RP04/5/6

FORMATTER
CZRJBB0

AH-9187B-MC
COPYRIGHT © 74-77
FICHE 1 OF 1

JAN 1978
digital
MADE IN USA

.REM @

I D E N T I F I C A T I O N

PRODUCT CODE: AC-9185B-MC

PRODUCT NAME: CZRJBB0 RP04/5/6 FORMATTER PROGRAM

DATE CREATED: DECEMBER 1977

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: C. HESS

COPYRIGHT (C) 1974, 1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

THE INFORMATION IN THIS STATEMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

ACTUAL DISTRIBUTION OF THE SOFTWARE DESCRIBED IN THIS DOCUMENT WILL BE SUBJECT TO TERMS AND CONDITIONS TO BE ANNOUNCED ON SOME FUTURE DATE BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A LICENSE TO USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DEC'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DEC.

CONTENTS

1. ABSTRACT

2. REQUIREMENTS

    2.1 EQUIPMENT
    2.2 PRELIMINARY PROGRAMS

3. LOADING PROCEDURES

4. STARTING PROCEDURES

    4.1 STARTING ADDRESSES
    4.2 OPERATOR ACTION
    4.3 RH11 - RH70 UNIBUS ADDRESS
    4.4 OTHER UNIBUS ADDRESSES

5. SWITCH REGISTER SETTINGS

6. ERROR MESSAGES

7. MISCELLANEOUS

    7.1 FORMAT TIMES
    7.2 HALTING THE PROGRAM
    7.3 SURFACE VERIFICATION

8. PROGRAM DESCRIPTION

    8.1 FORMAT OPERATION
    8.2 CHECK OPERATION
    8.3 POSITIONER VERIFICATION

9. PROGRAM LISTING

1. ABSTRACT

THE RP04/5/6 FORMATTER PROGRAM FORMATS THE DISK PACK AND PERFORMS A CURSORY CHECK OF THE PACK'S SURFACE. THE PROGRAM ALLOWS THE OPERATOR TO SPECIFY ADDRESS LIMITS, PATTERNS, AND EITHER 16 BIT OR 18 BIT FORMAT MODE. THE PROGRAM VERIFIES EACH TRACK WRITTEN AS WELL AS VERIFYING THE FORMAT OPERATION.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
8K MEMORY
TELETYPE
PROGRAM LOAD DEVICE
KW11-L OR KW11-P CLOCK
RH11 OR RH70 WITH 1 - 8 RP04, RP05, RP06 DISK DRIVES

2.2 PRELIMINARY PROGRAMS

RP04/5/6 DISKLESS CONTROLLER TEST
PART 1 (MAINDEC-11-DZRJG)
PART 2 (MAINDEC-11-DZRJH)

RP04/5/6 FUNCTIONAL CONTROLLER TEST
PART 1 (MAINDEC-11-DZRJI)
PART 2 (MAINDEC-11-DZRJJ)

3. LOADING PROCEDURES

THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA USING THE ASSOCIATED LOADER. THE PROGRAM MAY NOT BE INCLUDED IN AN 'XXDP' CHAIN.

4. STARTING PROCEDURES

4.1 STARTING ADDRESSES

THE PROGRAM IS STARTED FROM LOCATION 200(8) IF THE ADDRESS OF THE RH11 OR RH70 WILL NOT BE CHANGED

THE PROGRAM IS STARTED FROM LOCATION 204(8) IF THE ADDRESS OF THE RH11 OR RH70 IS TO BE CHANGED FROM THE PRELOADED VALUE. (SEE SECTION 4.3)

4.2 OPERATION ACTION

1. LOAD THE PROGRAM INTO MEMORY (SEE SECTION 3).

2. LOAD THE STARTING ADDRESS - 200(8) OR 204(8).

3. SET THE SWITCHES AS REQUIRED AND PRESS 'START'.

   IF THIS IS THE PROGRAM'S FIRST START, THE STATUS OF THE DRIVES ON THE SELECTED MASSBUS SUBSYSTEM WILL BE TYPED OUT. THIS TYPEOUT MAY BE INHIBITED ON SUBSEQUENT STARTS BY SETTING SW<02>.

4. THE PROGRAM WILL THEN TYPE THE FOLLOWING MESSAGE:

   'PROGRAM MODE (C OR F):'

   ENTER THE APPROPRIATE CODE: 'C' FOR 'CHECK' OPERATION OR 'F' FOR 'FORMAT & VERIFY'. IF A 'CARRIAGE RETURN' IS ENTERED IN RESPONSE TO THE REQUEST, THE PROGRAM WILL ASSUME 'FORMAT & VERIFY'.

5. THE PROGRAM WILL THEN ASK FOR THE FORMATTING MODE:

   'OPERATE IN 22 SECTOR (16 BIT) MODE (Y OR N)'

   ENTER THE APPROPRIATE CHARACTER: 'Y' FOR 16 BIT MODE OR 'N' FOR 18 BIT MODE. IF A 'CARRIAGE RETURN' IS ENTERED IN RESPONSE TO THIS REQUEST, THE PROGRAM WILL ASSUME 16 BIT MODE.

6. THE PROGRAM WILL THEN ASK FOR A DRIVE:

   'DRIVE: '

   ENTER THE ADDRESS OF THE DRIVE TO BE FORMATTED. A 'PERIOD' OR 'CARRIAGE RETURN' ENTRY WILL SELECT DRIVE 0. IF THE DRIVE SELECTED IS NOT AVAILABLE, THE PROGRAM WILL TYPE AN ERROR MESSAGE AND RETURN TO THE DRIVE ADDRESS REQUEST.

7. AFTER THE OPERATOR HAS SELECTED A DRIVE, THE PROGRAM WILL ASK FOR ADDRESS LIMITS FOR THE SELECTED DRIVE:

   'ENTER ADDRESS LIMITS: '

   THE PREVIOUSLY SELECTED OR DEFAULT VALUES FOR BEGINNING CYLINDER AND TRACK AND FOR ENDING CYLINDER AND TRACK WILL BE TYPED OUT. IF A 'CARRIAGE RETURN' IS TYPED AS A RESPONSE, THE PRESENT VALUE WILL BE USED; IF A 'PERIOD' IS TYPED, THE PROGRAM WILL BYPASS THE REMAINING ENTRIES AND WILL USE THEIR PRESENT VALUES. NOTE THAT THE CYLINDER AND TRACK VALUES ARE D E C I M A L NUMBERS. THE ADDRESS SPECIFIED BY THE BEGINNING CYLINDER AND TRACK MUST BE LESS THAN THE ADDRESS SPECIFIED BY THE ENDING CYLINDER AND TRACK ADDRESS.

7. THE PROGRAM WILL THEN ASK FOR THE DATA PATTERN:

```
'SELECT DATA PATTERN
  (0) ZERO'S
  (1) ONES
  (2) WORST CASE:'
```

ENTER THE CODE FOR THE REQUIRED PATTERN. 'CARRIAGE RETURN' OR 'PERIOD' ENTRIES WILL CAUSE THE PROGRAM TO USE THE 'WORST CASE' PATTERN.

THE 'WORST CASE' PATTERN IS THE FOLLOWING OCTAL SEQUENCE REPEATED THROUGH THE DATA AREA OF THE SECTOR:

```
165555
133333
```

8. THE PROGRAM WILL THEN TYPE:

'STARTING FORMAT ON DRIVE N'

OR

'STARTING CHECK ON DRIVE N'

9. THE OPERATOR CAN DETERMINE WHERE THE DRIVE IS DURING THE FORMAT OR CHECK OPERATION BY TYPING A 'CONTROL C'. THE PROGRAM WILL TYPE THE FOLLOWING MESSAGE:

'PRESENT ADDRESS IS: CXXX TXX'

IF A 'CONTROL C' IS TYPED WHILE THE PROGRAM IS TYPING THE CURRENT ADDRESS, THE PROGRAM WILL ABORT THE FORMAT (OR CHECK) OPERATION AND RETURN TO THE 'DRIVE' REQUEST.

10. IF THE OPERATOR DOES NOT SELECT A DIFFERENT DRIVE WHEN THE FORMAT OR CHECK OPERATION HAS COMPLETED, THE PROGRAM WILL NOT ALTER THE ADDRESS LIMITS SPECIFIED AT THE START OF THE PREVIOUS OPERATION. IF THE FORMAT OR CHECK OPERATION IS TERMINATED BY A 'CONTROL C' OR IF A DIFFERENT DRIVE IS SELECTED, THE PROGRAM WILL RESET THE ADDRESS LIMITS TO THE VALUES APPROPRIATE FOR THE DRIVE TYPE.

11. TO CHANGE EITHER THE ADDRESSING MODE OR THE OPERATION MODE, THE PROGRAM MUST BE STARTED FROM LOCATION 200(8) OR 204(8) AGAIN.

4.3 RH11 - RH70 UNIBUS ADDRESS

THE PROGRAM ASSUMES THAT THE RH11 OR RH70 ADDRESSES START AT 176700 AND THAT THE VECTOR ADDRESS IS 254. THESE ADDRESSES MAY BE CHANGED WHEN THE PROGRAM IS STARTED FROM LOCATION 204(8). IF THE RH11 - RH70 IS NOT AT THE DEFAULT ADDRESS, THE PROGRAM MUST BE STARTED FROM 204(8) INITIALLY AS THE PROGRAM GOES THROUGH THE ADDRESS CHANGE ROUTINE AT INITIAL START ONLY. ENTER THE RH11 RH70 ADDRESS IN RESPONSE TO THE REQUEST FROM THE PROGRAM.

4.4 OTHER UNIBUS ADDRESSES

LOC	TAG	CONTENTS	FUNCTION
1144	$TKS	177560	TTY KEYBOARD STATUS REGISTER
1146	$TKB	177562	TTY KEYBOARD BUFFER REGISTER
1150	$TPS	177564	TTY PRINTER STATUS REGISTER
1152	$TPB	177566	TTY PRINTER BUFFER REGISTER
1176	$LKCSR	172540	KW11-P CONTROL REGISTER
1200	$LKCSB	172542	KW11-P COUNTER REGISTER
1202	$LPVEC	104	KW11-P VECTOR ADDRESS
1206	$LKS	177546	KW11-L CONTROL REGISTER
1210	$LKV	100	;ADDRESS OF KW11-L VECTOR

5. SWITCH REGISTER SETTINGS

```
SW<15>=1...HALT ON ERROR
SW<13>=1...INHIBIT ERROR TYPEOUTS
SW<10>=1...BELL ON ERROR
SW<09>=1...LOOP ON ERROR
SW<02>=1...DON'T DISPLAY SYSTEM STATUS AFTER INITIAL START
SW<01>=1...LOOP ON THE CURRENT TRACK
SW<00>=1...LOOP THE PROGRAM ON THE SELECTED DRIVE
```

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS AT A HIGHER PRIORITY PROCESSING AN RP04/5/6 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

```
'SWR = NNNNNN    NEW ='
```

EACH TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

6. ERROR MESSAGES

1. 'RH11 INTERRUPT OCCURRED (RPAS=0) - AN INTERRUPT OCCURRED, BUT NOTHING ON THE MASSBUS IS INDICATING AN ATTENTION.

2. 'UNEXPECTED ATTENTION OCCURRED' - THE INDICATED DRIVE INTERRUPTED, BUT NO INTERRUPT WAS EXPECTED FROM THE INDICATED DRIVE.

3. 'MASSBUS PARITY ERROR (MCPE=1)' - A CONTROL BUS PARITY ERROR WAS DETECTED BY THE RH11 WHEN THE INDICATED REGISTER WAS READ.

4. 'MASSBUS PARITY ERROR (PAR=1)' - A CONTROL BUS PARITY ERROR OCCURRED WHEN THE INDICATED REGISTER WAS WRITTEN.

5. 'ADDRESS PLUG CHANGE BIT SET' - THE PROGRAM FOUND THE 'OPE' BIT SET FOR THE INDICATED DRIVE.

6. 'RH11 DIDN'T RESPOND TO ADDRESSING' - THE PROGRAM ACCESSED THE RH11 AT THE INDICATED ADDRESS AND RECEIVED NO RESPONSE.

7. 'DRIVE OFFLINE' - THE INDICATED DRIVE HAS GONE OFFLINE

8. 'PERSISTENT DRIVE UNSAFE ERROR' - THE INDICATED DRIVE HAS BECOME UNSAFE AND THE CONDITION CANNOT BE CLEARED BY ISSUING 'DRIVE CLEAR' INSTRUCTIONS.

9. 'UNCORRECTABLE MASSBUS PARITY ERROR' - THE PROGRAM ATTEMPTED TO PERFORM AN OPERATION AND DETECTED 3 SUCESSIVE MASSBUS PARITY ERRORS OR THE PROGRAM ATTEMPTED TO CLEAR A 'PAR' ERROR IN THE DRIVE AND A PARITY ERROR OCCURRED.

10. 'SOFTWARE TIMEOUT' - THE OPERATION FAILED TO COMPLETE WITHIN 1 SECOND.

11. 'DRIVE UNSAFE ERROR' - A NON-PERSISTENT UNSAFE ERROR OCCURRED DURING THE OPERATION.

12. 'CONTROLLER/DRIVE ERROR DURING WRITE' - THE INDICATED NON-DATA ERROR WAS DETECTED DURING A FORMAT OPERATION.

13. 'CONTROLLER/DRIVE ERROR DURING WRITE CHECK' - A NON-DATA ERROR OCCURRED DURING THE WRITE CHECK.

14. 'DATA ERROR DURING WRITE CHECK' - A DATA RELATED ERROR OCCURRED DURING A WRITE CHECK OPERATION. A DATA ERROR IS CONSIDERED TO BE ONE OF THE FOLLOWING ERRORS: 'DCK', 'OPI', 'DTE', 'HCRC', 'HCE', OR 'FER'.

15. 'RETRIES NOT SUCESSFUL - SECTOR NOT ACCEPTABLE' - THE INDICATED SECTOR FAILED DURING RETRY FOLLOWING A DATA ERROR.

16. 'CONTROLLER/DRIVE ERROR VERIFYING HEADERS' - AN ERROR OCCURRED DURING THE VERIFICATION PASS AFTER FORMATTING.

17. ''HCE' ERROR VERIFYING HEADERS' - A HEADER ERROR OCCURRED DURING

THE VERIFICATION PASS AFTER FORMATTING.

18. 'CYLINDER FIELD IN HEADER IS NOT CORRECT' - THE CYLINDER FIELD FROM A HEADER READ DURING THE VERIFICATION PASS IS NOT CORRECT.

19. 'WRITE CHECK ERROR' - THE RH11 REPORTED A WRITE CHECK ERROR AND NO DRIVE ERRORS WERE SET.

7. MISCELLANEOUS

7.1 FORMAT TIME

IT TAKES APPROXIMATELY 8 MINUTES TO FORMAT AN ENTIRE RP04/5 PACK AND APPROXIMATELY 16 MINUTES TO FORMAT AN RP06 PACK. THE 'CHECK' MODE TIME FOR AN ENTIRE PACK IS 4 MINUTES FOR RP04/5'S AND 8 MINUTES FOR RP06'S.

7.2 HALTING THE PROGRAM

THE OPERATOR SHOULD NOT HALT THE PROGRAM DURING A FORMAT OPERATION. HALTING THE PROGRAM MAY LEAVE A SECTOR INCORRECTLY FORMATTED. TO TERMINATE THE FORMAT, TYPE A 'CONTROL C' AND WHILE THE PROGRAM IS TYPING THE ADDRESS, TYPE ANOTHER 'CONTROL C'; THIS SEQUENCE RETURNS THE PROGRAM TO THE DRIVE ADDRESS ENTRY ROUTINE.

7.3 SURFACE VERIFICATION

THE FORMATTER PROGRAM IS NOT INTENDED TO BE USED TO PERFORM DISK PACK VERIFICATION. IF THE PROGRAM REPORTS A SECTOR AS BEING 'NOT ACCEPTABLE', THIS MAY IN FACT INDICATE A BAD SURFACE; HOWEVER, SECTORS WHICH 'PASSED' MAY OR MAY NOT BE GOOD.

8. PROGRAM DESCRIPTION

8.1 FORMAT OPERATION

THE PROGRAM FORMATS THE PACK ONE TRACK AT A TIME BETWEEN THE LIMITS SPECIFIED BY THE DRIVE.

THE FORMAT OPERATION CONSISTS OF A WRITE HEADER AND DATA COMMAND FOR THE ENTIRE TRACK FOLLOWED BY A WRITE CHECK HEADER AND DATA COMMAND. IF AN ERROR OCCURS DURING THE WRITE, THE PROGRAM WILL RETRY THE WRITE BEFORE CONTINUING. IF A DATA RELATED ERROR IS DETECTED DURING THE WRITE CHECK (A DATA ERROR IS DEFINED AS ONE OF THE FOLLOWING ERRORS: 'DCK', 'OPI', 'DTE', 'HCRC', 'HCE' OR 'FER'), THE SECTOR IN ERROR WILL BE REWRITTEN. THE PROGRAM WILL CHECK THE SECTOR TWICE; IF A DATA ERROR IS DETECTED IN THE SECTOR DURING EITHER OF THE WRITE CHECKS, THE PROGRAM WILL DECLARE

THE SECTOR AS BEING 'UNACCEPTABLE'. FOLLOWING THIS SEQUENCE,
THE REMAINDER OF THE TRACK IS CHECKED. IF DATA ERRORS ARE ENCOUNTERED
IN ANY OF THE REMAINING SECTORS, EACH SECTOR WILL BE HANDLED AS
DESCRIBED ABOVE.

IF A NON-DATA RELATED ERROR OCCURS DURING THE WRITE CHECK, THE
PROGRAM WILL RETRY THE COMMAND FOR THE ENTIRE TRACK BEFORE PROCEEDING
TO THE NEXT TRACK.

IT SHOULD BE NOTED THAT THE FORMATTER PROGRAM FORMATS THE PACK ONLY
AND IS NOT DIRECTLY VERIFYING THE SURFACE OF THE PACK. ERRORS
THAT ARE ENCOUNTERED MAY BE THE RESULT OF BAD AREAS ON THE PACK
BUT, AS THE PROGRAM IS NOT DESIGNED TO PERFORM AN EXHAUSTIVE CHECK
OF THE DISK PACK SURFACE, THE PROGRAM CANNOT MAKE THIS CONCLUSION.
IN GENERAL, HOWEVER, THE OPERATOR CAN ASSUME THAT SECTORS WHICH
THE PROGRAM CALLS 'UNACCEPTABLE' INDICATE BAD AREAS OF THE PACK;
UNFORTUNATELY, SECTORS WHICH 'PASS' CANNOT BE ASSUMED TO BE GOOD.

DURING THE FORMAT OPERATION, THE PROGRAM FILLS THE DATA FIELD
WITH THE PATTERN SELECTED BY THE OPERATOR. THE KEYWORDS IN THE
HEADER ARE ALWAYS SET TO ZERO.

8.2 CHECK OPERATION

THE CHECK OPERATION IS IDENTICAL TO THE WRITE CHECK PORTION OF
THE FORMAT OPERATION OPERATION DESCRIBED IN SECTION 8.1 EXCEPT
THAT THE PROGRAM WILL NOT RE-WRITE ERROR SECTORS.

8.3 POSITIONER VERIFICATION

AFTER THE PROGRAM COMPLETES THE FORMAT OPERATION, THE
POSITIONER IS RETURNED TO THE STARTING CYLINDER AND THE HEADER
FROM SECTOR 0 ON THE STARTING TRACK IS READ. THE CYLINDER ADDRESS
FIELD FROM THE HEADER IS COMPARED TO THE REQUESTED CYLINDER; IF
THE CYLINDER ADDRESSES DO NOT COMPARE, AN ERROR MESSAGE IS TYPED.
THE PROGRAM CHECKS THE CYLINDER ADDRESS FIELD FROM THE HEADER
OF SECTOR 0 ON THE BEGINNING TRACK OF EACH CYLINDER FORMATTED.
THIS CHECK IS PERFORMED TO CONFIRM THAT THE DISK'S POSITIONER
ADVANCED PROPERLY DURING THE FORMAT OPERATION.

9. PROGRAM LISTING

```
.TITLE  CZRJBB0, RP04/5/6 FMTR
;*COPYRIGHT (C) 1976,1977
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY C. HESS
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
;*
```

```
                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         2            DON'T DISPLAY SYSTEM STATUS AFTER INITIAL START
                                ;*         1            LOOP ON THE CURRENT TRACK
                                ;*         0            LOOP THE PROGRAM ON THE SELECTED DRIVE

                                .SBTTL  TRAP CATCHER

            000000                      .=0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
            000174                      .=174
000174  000000                 DISPREG: .WORD  0                  ;;SOFTWARE DISPLAY REGISTER
000176  000000                 SWREG:   .WORD  0                  ;;SOFTWARE SWITCH REGISTER

                                .SBTTL  ACT11 HOOKS

                                ;;*****************************************************************
                                ;HOOKS REQUIRED BY ACT11
            000200                      $SVPC=.                    ;SAVE PC
            000046                      .=46
000046  005174                          $ENDAD                     ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
            000052                      .=52
000052  020000                          .WORD   20000              ;;2)SET LOC.52 TO 20000
            000200                      .=$SVPC                    ;; RESTORE PC

                                .SBTTL  STARTING ADDRESS = 200
            000200                      .=200
000200  000137  002076                  JMP     BEGIN1             ;NORMAL STARTING ADDRESS

                                .SBTTL  STARTING ADDRESS TO CHANGE THE RH11 ADDRESS = 204
000204  000137  002066                  JMP     BEGIN              ;CHANGE THE RH11 ADDRESS

                                .SBTTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
            001100              STACK=  1100
                                .EQUIV  EMT,ERROR          ;;BASIC DEFINITION OF ERROR CALL
                                .EQUIV  IOT,SCOPE          ;;BASIC DEFINITION OF SCOPE CALL

                                ;*MISCELLANEOUS DEFINITIONS
            000011              HT=     11                 ;;CODE FOR HORIZONTAL TAB
            000012              LF=     12                 ;;CODE FOR LINE FEED
            000015              CR=     15                 ;;CODE FOR CARRIAGE RETURN
            000200              CRLF=   200                ;;CODE FOR CARRIAGE RETURN-LINE FEED
            177776              PS=     177776             ;;PROCESSOR STATUS WORD
                                .EQUIV  PS,PSW
            177774              STKLMT= 177774             ;;STACK LIMIT REGISTER
```

```
                        
000534  177772          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
000535  177570          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
000536  177570          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
000537
000538                  ;*GENERAL PURPOSE REGISTER DEFINITIONS
000539  000000          R0=     %0              ;;GENERAL REGISTER
000540  000001          R1=     %1              ;;GENERAL REGISTER
000541  000002          R2=     %2              ;;GENERAL REGISTER
000542  000003          R3=     %3              ;;GENERAL REGISTER
000543  000004          R4=     %4              ;;GENERAL REGISTER
000544  000005          R5=     %5              ;;GENERAL REGISTER
000545  000006          R6=     %6              ;;GENERAL REGISTER
000546  000007          R7=     %7              ;;GENERAL REGISTER
000547  000006          SP=     %6              ;;STACK POINTER
000548  000007          PC=     %7              ;;PROGRAM COUNTER
000549
000550                  ;*PRIORITY LEVEL DEFINITIONS
000551  000000          PR0=    0               ;;PRIORITY LEVEL 0
000552  000040          PR1=    40              ;;PRIORITY LEVEL 1
000553  000100          PR2=    100             ;;PRIORITY LEVEL 2
000554  000140          PR3=    140             ;;PRIORITY LEVEL 3
000555  000200          PR4=    200             ;;PRIORITY LEVEL 4
000556  000240          PR5=    240             ;;PRIORITY LEVEL 5
000557  000300          PR6=    300             ;;PRIORITY LEVEL 6
000558  000340          PR7=    340             ;;PRIORITY LEVEL 7
000559
000560                  ;*"SWITCH REGISTER" SWITCH DEFINITIONS
000561  100000          SW15=   100000
000562  040000          SW14=   40000
000563  020000          SW13=   20000
000564  010000          SW12=   10000
000565  004000          SW11=   4000
000566  002000          SW10=   2000
000567  001000          SW09=   1000
000568  000400          SW08=   400
000569  000200          SW07=   200
000570  000100          SW06=   100
000571  000040          SW05=   40
000572  000020          SW04=   20
000573  000010          SW03=   10
000574  000004          SW02=   4
000575  000002          SW01=   2
000576  000001          SW00=   1
000577                  .EQUIV  SW09,SW9
000578                  .EQUIV  SW08,SW8
000579                  .EQUIV  SW07,SW7
000580                  .EQUIV  SW06,SW6
000581                  .EQUIV  SW05,SW5
000582                  .EQUIV  SW04,SW4
000583                  .EQUIV  SW03,SW3
000584                  .EQUIV  SW02,SW2
000585                  .EQUIV  SW01,SW1
000586                  .EQUIV  SW00,SW0
000587
000588                  ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
000589  100000          BIT15=  100000
```

```
                        040000          BIT14=  40000
                        020000          BIT13=  20000
                        010000          BIT12=  10000
                        004000          BIT11=  4000
                        002000          BIT10=  2000
                        001000          BIT09=  1000
                        000400          BIT08=  400
                        000200          BIT07=  200
                        000100          BIT06=  100
                        000040          BIT05=  40
                        000020          BIT04=  20
                        000010          BIT03=  10
                        000004          BIT02=  4
                        000002          BIT01=  2
                        000001          BIT00=  1
                                        .EQUIV  BIT09,BIT9
                                        .EQUIV  BIT08,BIT8
                                        .EQUIV  BIT07,BIT7
                                        .EQUIV  BIT06,BIT6
                                        .EQUIV  BIT05,BIT5
                                        .EQUIV  BIT04,BIT4
                                        .EQUIV  BIT03,BIT3
                                        .EQUIV  BIT02,BIT2
                                        .EQUIV  BIT01,BIT1
                                        .EQUIV  BIT00,BIT0

                                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                        000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                        000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                        000014          TBITVEC=14              ;;"T" BIT
                        000014          TRTVEC= 14              ;;TRACE TRAP
                        000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                        000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                        000024          PWRVEC= 24              ;;POWER FAIL
                        000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                        000034          TRAPVEC=34              ;;"TRAP" TRAP
                        000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
                        000064          TPVEC=  64              ;;TTY PRINTER VECTOR
                        000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR

                        ;;*****************************************************************

                        .SBTTL  RH11 REGISTERS

                        ;;*****************************************************************

                        ;CONTROL AND STATUS REGISTER 1 (RPCS1)

                        000100          IE=     100             ;INTERRUPT ENABLE (BIT #6)
                        000200          RDY=    200             ;READY (BIT #7)
                        000400          A16=    400             ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
                        001000          A17=    1000            ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
                        002000          PSEL=   2000            ;PORT SELECT (BIT #10)
                        020000          MCPE=   20000           ;MASSBUSS PARITY ERROR (BIT #13)
                        040000          TRE=    40000           ;TRANSFER ERROR (BIT #14)
                                        ;SC=    100000          ;SPECIAL CONDITION (BIT #15)
```

```
                        ;WORD COUNT REGISTER (RPWC)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        ;BUS ADDRESS REGISTER (RPBA)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

                        ;CONTROL AND STATUS REGISTER 2 (RPCS2)

000001                  US1=    1               ;UNIT SELECT (BIT #0)
000002                  US2=    2               ;UNIT SELECT (BIT #1)
000004                  US4=    4               ;UNIT SELECT (BIT #2)
000010                  BAI=    10              ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
000020                  PAT=    20              ;MASSBUS PARITY TEST (BIT #4)
000040                  CLR=    40              ;CLEAR (BIT #5)
000100                  IR=     100             ;INPUT READY (BIT #6)
000200                  OR=     200             ;OUTPUT READY (BIT #7)
000400                  MPE=    400             ;MASS BUS PARITY ERROR (BIT #8)
001000                  MXF=    1000            ;MISSED TRANSFER ERROR (BIT #9)
002000                  PGE=    2000            ;PROGRAM ERROR (BIT #10)
004000                  NEM=    4000            ;NON EXISTENT MEMORY (BIT #11)
010000                  NED=    10000           ;NON EXISTENT DRIVE (BIT #12)
020000                  UPE=    20000           ;UNIBUS PARITY ERROR (BIT #13)
040000                  WCE=    40000           ;WRITE CHECK ERROR (BIT #14)
100000                  DLT=    100000          ;DATA LATE (BIT #15)

                        ;DATA BUFFER REGISTER (RPDB)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)


                        ;;*****************************************************************

                        .SBTTL  RP04/5/6 REGISTERS

                        ;;*****************************************************************

                        ;CONTROL AND STATUS 1 REGISTER. (#00)

000001                  GO=     1               ;GO BIT (BIT #0)
000002                  F1=     2               ;FUNCTION CODE BIT #1
000004                  F2=     4               ;FUNCTION CODE BIT #2
000010                  F3=     10              ;FUNCTION CODE BIT #3
000020                  F4=     20              ;FUNCTION CODE BIT #4
000040                  F5=     40              ;FUNCTION CODE BIT #5
004000                  DVA=    4000            ;DEVICE AVAILABLE (BIT #11)

                        ;DRIVE STATUS REGISTER (RPDS1) (#01)

                        ;DFS=   1               DRIVE FORWARD 5"/SEC.  (BIT #0)
000002                  DFF20=  2               ;DRIVE FORWARD 20"/SEC.  (BIT #1)
000004                  DIGB=   4               ;DRIVE TO INNER GUARD BAND (BIT #2)
000010                  GRV=    10              ;GO REVERSE (BIT #3)
000020                  DL64=   20              ;DIFFERENCE LESS THAN 64 (BIT #4)
000040                  DE1=    40              ;DIFFERENCE EQUALS 1 (BIT #5)
000100                  VV=     100             ;VOLUME VALID (BIT #6)
000200                  DRY=    200             ;DRIVE READY (BIT #7)
```

```
                        
000400                  DPR=    400             ;DRIVE PRESENT (BIT #8)
001000                  PGM=    1000            ;PROGRAMABLE (BIT #9)
002000                  LST=    2000            ;LAST SECTOR TRANSFERRED (BIT #10)
004000                  WRL=    4000            ;WRITE LOCK (BIT #11)
010000                  MOL=    10000           ;MEDIUM ON-LINE (BIT #12)
020000                  PIP=    20000           ;POSITIONING OPERATION IN PROGRESS (BIT #13)
040000                  ERR=    40000           ;COMPOSITE ERROR (BIT #14)
100000                  ATA=    100000          ;ATTENTION ACTIVE (BIT #15)
                        
                        ;ERROR REGISTER #01 (RPER1) (#02)
                        
000001                  ILF=    1               ;ILLEGAL FUNCTION (BIT #0)
000002                  ILR=    2               ;ILLEGAL REGISTER (BIT #1)
000004                  RMR=    4               ;REGISTER MODIFICATION REFUSED (BIT #2)
000010                  PAR=    10              ;PARITY ERROR (BIT #3)
000020                  FER=    20              ;FORMAT ERROR (BIT #4)
000040                  WCF=    40              ;WRITE CLOCK FAIL (BIT #5)
000100                  ECH=    100             ;ECC HARD ERROR (BIT #6)
000200                  HCE=    200             ;HEADER COMPARE ERROR (BIT #7)
000400                  HCRC=   400             ;HEADER CRC ERROR (BIT #8)
001000                  AOE=    1000            ;ADDRESS OVERFLOW ERROR (BIT #9)
002000                  IAE=    2000            ;INVALID ADDRESS ERROR (BIT #10)
004000                  WLE=    4000            ;WRITE LOCK ERROR (BIT #11)
010000                  DTE=    10000           ;DRIVE TIMING ERROR (BIT #12)
020000                  OPI=    20000           ;OPERATION INCOMPLETE (BIT #13)
040000                  UNS=    40000           ;DRIVE UNSAFE (BIT #14)
100000                  DCK=    100000          ;DATA CHECK ERROR (BIT 15)
                        
                        ;MAINTAINABILITY REGISTER (RPMR)(#03)
                        
000001                  DMD=    1               ;DIAGINOSTIC MODE (BIT #0)
000002                  MCLK=   2               ;MAINTAINABILITY CLOCK (BIT #1)
000004                  MINX=   4               ;MAINTAINABILITY INDEX (BIT #2)
000010                  MSTCK=  10              ;MAINTAINABILITY SECTOR CLOCK (BIT #3)
000020                  MRD=    20              ;MAINTAINABILITY READ (BIT #4)
000040                  MWR=    40              ;MAINTAINABILITY WRITE (BIT #5)
000200                  DTSY=   200             ;MAINTAINABILITY SYNC DETECTED (BIT #7)
                        
                        ;ATTENTION SUMMARY PSEUDO-REGISTER (RPAS) (#04)
                        
000001                  AT0=    1               ;DEVICE 0 (BIT #0)
000002                  AT1=    2               ;DEVICE 1 (BIT #1)
000004                  AT2=    4               ;DEVICE 2 (BIT #2)
000010                  AT3=    10              ;DEVICE 3 (BIT #3)
000020                  AT4=    20              ;DEVICE 4 (BIT #4)
000040                  AT5=    40              ;DEVICE 5 (BIT #5)
000100                  AT6=    100             ;DEVICE 6 (BIT #6)
000200                  AT7=    200             ;DEVICE 7 (BIT #7)
                        
                        ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)
                        ;(EACH BIT IS CALLED BY BIT NUMBER)
                        
                        ;DRIVE TYPE REGISTER (RPDT) (#06)
                        
000001                  DT00=   1               ;DRIVE TYPE NUMBER BIT 1
000002                  DT01=   2               ;DRIVE TYPE NUMBER BIT 2
```

```
                                ;DRIVE TYPE NUMBER BIT 3
000004          DT02=   4       ;DRIVE TYPE NUMBER BIT 4
000010          DT03=   10      ;DRIVE TYPE NUMBER BIT 5
000020          DT04=   20      ;DRIVE TYPE NUMBER BIT 6
000040          DT05=   40      ;DRIVE TYPE NUMBER BIT 7
000100          DT06=   100     ;DRIVE TYPE NUMBER BIT 8
000200          DT07=   200     ;DRIVE TYPE NUMBER BIT 9
000400          DT08=   400     ;DRIVE REQUEST REQUIRED (BIT #11)
004000          DRQ=    4000    ;MOVING HEAD (BIT #13)
020000          MOH=    20000   ;TAPE DRIVE (BIT #14)
040000          TAP=    40000   ;NOT BLOCK ADDRESSED (BIT #15)
100000          NBA=    100000

                ;LOOK-AHEAD REGISTER (RPLA) (#07)

000001          EXT1=   1       ;EXTENSION 1 (BIT #0)
000002          EXT2=   2       ;EXTENSION 2 (BIT #1)
000004          EXT4=   4       ;EXTENSION 3 (BIT #2)
000010          EXT10=  10      ;EXTENSION 4 (BIT #3)
000020          EXT20=  20      ;EXTENSION 5 (BIT #4)
000040          EXT40=  40      ;EXTENSION 6 (BIT #5)
000100          SC1=    100     ;SECTOR COUNT FIELD 0 (BIT #6)
000200          SC2=    200     ;SECTOR COUNT FIELD 1 (BIT #7)
                .SC4=   400     ;SECTOR COUNT FIELD 2 (BIT #8)
001000          SC10=   1000    ;SECTOR COUNT FIELD 3 (BIT #9)
002000          SC20=   2000    ;SECTOR COUNT FIELD 4 (BIT #10)
004000          TRK1=   4000    ;TRACK FIELD 1 (BIT #11)
010000          TRK2=   10000   ;TRACK FIELD 2 (BIT #12)
020000          TRK4=   20000   ;TRACK FIELD 3 (BIT #13)
040000          TRK10=  40000   ;TRACK FIELD 4 (BIT #14)
100000          TRK20=  100000  ;TRACK FIELD 5 (BIT #15)

                ;RP04 ERROR REGISTER #2 (RPER2) (#10)

000001          WCU=    1       ;WRITE CURRENT UNSAFE (BIT #0)
000002          CSF=    2       ;CURRENT SINK FAILURE (BIT #1)
000004          WSU=    4       ;WRITE SELECT UNSAFE (BIT #2)
000010          CSU=    10      ;CURRENT SWITCH UNSAFE (BIT #3)
000020          MSE=    20      ;MOTOR SEQUENCE ERROR (BIT #4)
000040          TDF=    40      ;TRANSITIONS DETECTOR FAILURE (BIT #5)
000100          TUF=    100     ;TRANSITIONS UNSAFE (BIT #6)
000200          FEN=    200     ;FAILSAFE ENABLED (BIT #7)
000400          WRU=    400     ;WRITE READY UNSAFE (BIT #8)
001000          MHS=    1000    ;MULTIPLE HEAD SELECT (BIT #9)
002000          NHS=    2000    ;NO HEAD SELECTION (BIT #10)
004000          IXE=    4000    ;INDEX ERROR (BIT #11)
010000          VU30=   10000   ;30VOLT UNSAFE (BIT #12)
020000          PLU=    20000   ;PLO UNSAFE (BIT #13)
100000          ACU=    100000  ;AC UNSAFE (BIT #15)

                ;RP05/6 ERROR REGISTER #02 (RPER2) (#10)

000001          WCU=    1       ;WRITE CURRENT UNSAFE (BIT #0)
000002          CSF=    2       ;CURRENT SINK FAILURE (BIT #1)
000004          WSU=    4       ;WRITE SELECT UNSAFE (BIT #2)
000010          CSU=    10      ;CURRENT SWITCH UNSAFE (BIT #3)
000020          RAW=    20      ;READ AND WRITE (BIT #4)
```

```
                        ;TRANSITIONS DETECTOR FAILURE (BIT #5)
000040          TDF=    40
                        ;TRANSITIONS UNSAFE (BIT #6)
000100          TUF=    100
                        ;ABNORMAL STOP (BIT #7)
000200          ABS=    200
                        ;WRITE READY UNSAFE (BIT #8)
000400          WRU=    400
                        ;MUTLTIPLE HEAD SELECT (BIT #9)
001000          MHS=    1000
                        ;NO HEAD SELECTION (BIT #10)
002000          NHS=    2000
                        ;INDEX ERROR (BIT #11)
004000          IXE=    4000
                        ;PLO UNSAFE (BIT #12)
020000          PLU=    20000

                ;OFFSET REGISTER (RPOF) (#11)

                        ;OFFSET 25 MICRO INCHES (BIT #0)
000001          OF25=   1
                        ;OFFSET 50 MICRO INCHES (BIT #1)
000002          OF50=   2
                        ;OFFSET 100 MICRO INCHES (BIT #2)
000004          OF100=  4
                        ;OFFSET 200 MICRO INCHES (BIT #3)
000010          OF200=  10
                        ;OFFSET 400 MICRO INCHES (BIT #4)
000020          OF400=  20
                        ;OFFSET 800 MICRO INCHES (BIT #5)
000040          OF800=  40
                        ;OFFSET NEGATIVE (REVERSE) (BIT #5)
000200          OFREV=  200
                        ;HEADER COMPARE INHIBIT (BIT #10)
002000          HCI=    2000
                        ;ERROR CORRECTION CODE INHIBIT (BIT #11)
004000          ECI=    4000
                        ;FORMAT BIT (BIT #12
010000          FMT22=  10000

                ;DESIRED CYLINDER ADDRESS (RPCA) (#12)
                ;(EACH BIT IS CALLED BY BIT NUMBER)

                ;CURRENT CYLINDER ADDRESS (RPCC) (#13)
                ;(EACH BIT IS CALLED BY BIT NUMBER)

                ;SERIAL NUMBER REGISTER (RPSN) (#14)
                ;(EACH IS CALLED BY BIT NUMBER)

                ;RP04 ERROR REGISTER #03 (RPER3) (#15)

                        ;PACK SPEED UNSAFE (BIT #0)
000001          PSU=    1
                        ;VELOCITY UNSAFE (BIT #1)
000002          VUF=    2
                        ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
000010          UWR=    10
                        ;AC LOW (BIT #5)
000040          ACL=    40
                        ;DC LOW (BIT #6)
000100          DCL=    100
                        ;SEEK INCOMPLETE (BIT #14)
040000          SKI=    40000
                        ;OFF CYLINDER (BIT #15)
100000          OCYL=   100000

                ;RP05/6 ERROR REGISTER #03 (RPER3) (#15)

                        ;DC UNSAFE (BIT #0)
000001          DCU=    1
                        ;WRITE AND OFFSET (BIT #1)
000002          WAO=    2
                        ;AC LOW (BIT #5)
000040          ACL=    40
                        ;DC LOW (BIT #6)
000100          DCL=    100
                        ;OPERATOR PLUG ERROR (BIT #13)
020000          OPE=    20000
                        ;SEEK INCOMPLETE (BIT #14)
040000          SKI=    40000
                        ;OFF CYLINDER ERROR (BIT #15)
100000          OCYL=   100000

                ;ECC POSITION REGISTER (RPEC1) (#16)
                ;(EACH BIT IS CALLED BY BIT NUMBER)

                ;ECC PATTERN REGISTER (RPEC2) (#17)
                ;(EACH BIT IS CALLED BY BIT NUMBER)
```

```
                        ;;*****************************************************************

                        .SBTTL  RP04/5/6 DRIVER COMMANDS

                        ;;*****************************************************************

000101                  RNOP    =       101             ;NO OPERATION
000103                  UNLOAD  =       103             ;UNLOAD
000105                  SEEK    =       105             ;SEEK
000107                  RECAL   =       107             ;RECALIBRATE
000111                  DRVCLR  =       111             ;DRIVE CLEAR
000113                  RELSE   =       113             ;RELEASE
000115                  OFFSET  =       115             ;OFFSET
000117                  RTC     =       117             ;RETURN TO CENTER LINE
000121                  READIN  =       121             ;READ IN PRESET
000123                  ACK     =       123             ;PACK ACKNOWLEDGE
000131                  SEARCH  =       131             ;SEARCH
000141                  GETREG  =       141             ;GET REGISTERS
000143                  SETFMT  =       143             ;SET FORMAT (& ECI OR HCI)
000145                  SELDRV  =       145             ;SELECT DRIVE
000151                  WCKD    =       151             ;WRITE CHECK DATA
000153                  WCKHD   =       153             ;WRITE CHECK HEADER & DATA
000161                  WRTDAT  =       161             ;WRITE DATA
000163                  WRTHD   =       163             ;WRITE HEADER & DATA
000171                  RDDAT   =       171             ;READ DATA
000173                  RDHD    =       173             ;READ HEADER & DATA
```

```
                                .SBTTL  COMMON TAGS

                                ;;*****************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.

                                        .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001162  177607  000377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001166  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001167  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001170  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
000015                  CR      =       15
000012                  LF      =       12
001172  176700          $RPADR: .WORD   176700          ;RH11/RP04/5/6 UNIBUS ADDRESS
001174  000254          $RPVEC: .WORD   254             ;RH11 INTERRUPT VECTOR
001176  172540          $LKCSR: .WORD   172540          ;ADDRESS OF KW11-P CSR
001200  172542          $LKCSB: .WORD   172542          ;ADDRESS OF KW11-P COUNTER BUFFER
001202  000104  000106  $LPVEC: .WORD   104,106         ;ADDRESS OF KW11-P VECTOR
001206  177546          $LKS:   .WORD   177546          ;ADDRESS OF KW11-L CONTROL REGISTER
001210  000100  000102  $LLVEC: .WORD   100,102         ;ADDRESS OF KW11-L VECTOR
001214  000000          DRIVE:  .WORD   0               ;CONTAINS DRIVE NUMBER SELECTED
001216  000000          SOFSW:  .WORD   0               ;CONTENTS ARE FOR SOFTWARE DECISIONS
001220  000000          MODE:   .WORD   0               ;'FORMAT & VERIFY' OR 'CHECK' MODE INDICATOR
001222  000632          ENDCYL: .WORD   410.            ;ENDING CYLINDER
```

```
                                BEGCYL: .WORD   0               ;STARTING CYLINDER
                                ENDTRK: .WORD   18.             ;ENDING TRACK
                                BEGTRK: .WORD   0               ;STARTING TRACK
                                TTRKS:  .WORD   0               ;TOTAL # OF TRACKS TO BE FORMATTED
                                TTRKSC: .WORD   0               ;TOTAL # OF TRACKS COUNTER
                                TRKCNT: .WORD   0               ;COUNTS TRKS FROM 0-18 PER CYL
                                PATSEL: .WORD   0               ;CONTAINS PATTERN SELECTED
                                PATA:   .WORD   0               ;1ST WORD OF PATTERN
                                PATB:   .WORD   0               ;2ND WORD OF PATTERN
                                CYLCK:  .WORD   0               ;STORE CYLINDER ADDRESS FOR FORMAT VERIFICATION
                                RETRY:  .WORD   0               ;MAINTAINS # OF WRITE RETRIES MADE
                                SAVSEC: .WORD   0               ;CONTAINS LAST BAD SECTOR ON FORMAT
                                SAVWC:  .WORD   0               ;CONTAINS WC FOR REMAINING SECTORS ON ERROR
                                WC:     .WORD   0               ;20 OR 22 SECTOR TRACK SIZE (IN WORDS)
                                MWC:    .WORD   0               ;2'S COMPLEMENT OF 'WC'
                                HEDERR: .WORD   0               ;POSITIONING ERROR DURING FORMAT INDICATOR
                                SEC20:  .WORD   0               ;20 OR 22 SECTOR MODE INDICATOR
                                                                ;0 = 20 SECTOR MODE
                                                                ;1'S = 22 SECTOR MODE
                                MAXSEC: .WORD   0               ;MAXIMUM SECTOR ADDRESS (FOR EITHER 20 OR 22 SECTOR
                                                                ;FORMAT)
                                DS.CYL: .WORD   0               ;ADDRESS OF CURRENT CYLINDER
                                DS.TRK: .WORD   0               ;ADDRESS OF CURRENT TRACK
                                DDRIVE: .WORD   0               ;DRIVE NUMBER OF 'DRIVER' ERROR MESSAGES
                                ATTN:   .WORD   0               ;ATTENTION REGISTER IMAGE FOR 'DRIVER'
                                                                ;ERROR MESSAGES
                                CNTLC:  .WORD   0               ;ADDRESS OF '↑C' RETURN
                                CHGADR: .WORD   0               ;'CHANGE RH11 ADDRESS' FLAG

                                ;RH11/RP04/5/6 REGISTERS STORED HERE AFTER AN OPERATION

                                RP.REG: .WORD   0               ;RPCS1
                                        .WORD   0               ;RPWC
                                        .WORD   0               ;RPBA
                                        .WORD   0               ;RPDA
                                        .WORD   0               ;RPCS2
                                        .WORD   0               ;RPDS1
                                        .WORD   0               ;RPER1
                                        .WORD   0               ;RPAS
                                        .WORD   0               ;RPLA
                                        .WORD   0               ;RPDB
                                        .WORD   0               ;RPMR
                                        .WORD   0               ;RPDT
                                        .WORD   0               ;RPSN
                                        .WORD   0               ;RPOF
                                        .WORD   0               ;RPCA
                                        .WORD   0               ;RPCC
                                        .WORD   0               ;RPER2
                                        .WORD   0               ;RPER3
                                        .WORD   0               ;RPEC1
                                        .WORD   0               ;RPEC2


                                ;DATA/PARAMETER BLOCK - USED FOR ALL DRIVE OPERATION

                                FMTDPB: .BYTE   0               ;DRIVE NUMBER
```

```
1011  001355  000                             .BYTE   0               ;OFFSET VALUE OR FMT22,ECI, AND HCI
1012  001356  000                             .BYTE   0               ;COMMAND
1013  001357  000                             .BYTE   0               ;PSEL AND A17 AND A16
1014  001360  000000                          .WORD   0               ;WORD COUNT (NEG)
1015  001362  000000                          .WORD   0               ;BUFFER ADDRESS
1016  001364  000                             .BYTE   0               ;SECTOR ADDRESS
1017  001365  000                             .BYTE   0               ;TRACK ADDRESS
1018  001366  000000                          .WORD   0               ;CYLINDER ADDRESS
1019  001370  001304                          .WORD   RP.REG          ;ERROR TABLE POINTER
1020  001372  000000                          .WORD   0               ;STATUS-ERROR INDICATOR
1021                                                                   ;BIT 15 = 1: ERROR OCCURRED
1022                                                                   ;BIT 07 = 1: DONE
1023                                                                   ;BIT 14-10 AND BIT 06-03
1024                                                                   ;INDICATE TYPE OF ERROR
1025
1026
1027                                    ;PARAMETER POINTER TABLE
1028
1029  001374  001426  000000  001224    TABLE:  PAR1,0,BEGCYL
1030  001402  001441  000022  001230            PAR2,18.,BEGTRK
1031  001410  001454  000000  001222            PAR3,0,ENDCYL
1032  001416  001465  000022  001226            PAR4,18.,ENDTRK,0
1033  001424  000000
1034
1035                                    ;ASCII MESSAGES FOR ADDRESS PARAMETERS
1036
1037  001426  052123  051101  020124    PAR1:   .ASCIZ  @START CYL @
1038  001434  054503  020114  000
1039  001441  123     040524  052122    PAR2:   .ASCIZ  @START TRK @
1040  001446  052040  045522  000040
1041  001454  047105  020104  054503    PAR3:   .ASCIZ  @END CYL @
1042  001462  020114  000
1043  001465  105     042116  052040    PAR4:   .ASCIZ  @END TRK @
1044  001472  045522  000040
1045
1046
1047
1048  001476  025130                    ;SECTOR BUFFER ADDRESS TABLE
1049  001500  026140
1050  001502  027150    ADRTBL: .WORD   BUFP                     ;ADDRESS OF SECTOR 0
1051  001504  030160            .WORD   BUFP+<520.*1.>          ;ADDRESS OF SECTOR 1
1052  001506  031170            .WORD   BUFP+<520.*2.>          ;ADDRESS OF SECTOR 2
1053  001510  032200            .WORD   BUFP+<520.*3.>          ;ADDRESS OF SECTOR 3
1054  001512  033210            .WORD   BUFP+<520.*4.>          ;ADDRESS OF SECTOR 4
1055  001514  034220            .WORD   BUFP+<520.*5.>          ;ADDRESS OF SECTOR 5
1056  001516  035230            .WORD   BUFP+<520.*6.>          ;ADDRESS OF SECTOR 6
1057  001520  036240            .WORD   BUFP+<520.*7.>          ;ADDRESS OF SECTOR 7
1058  001522  037250            .WORD   BUFP+<520.*8.>          ;ADDRESS OF SECTOR 8
1059  001524  040260            .WORD   BUFP+<520.*9.>          ;ADDRESS OF SECTOR 9
1060  001526  041270            .WORD   BUFP+<520.*10.>         ;ADDRESS OF SECTOR 10
1061  001530  042300            .WORD   BUFP+<520.*11.>         ;ADDRESS OF SECTOR 11
1062  001532  043310            .WORD   BUFP+<520.*12.>         ;ADDRESS OF SECTOR 12
1063  001534  044320            .WORD   BUFP+<520.*13.>         ;ADDRESS OF SECTOR 13
1064  001536  045330            .WORD   BUFP+<520.*14.>         ;ADDRESS OF SECTOR 14
1065  001540  046340            .WORD   BUFP+<520.*15.>         ;ADDRESS OF SECTOR 15
1066  001542  047350            .WORD   BUFP+<520.*16.>         ;ADDRESS OF SECTOR 16
                                 .WORD   BUFP+<520.*17.>         ;ADDRESS OF SECTOR 17
                                 .WORD   BUFP+<520.*18.>         ;ADDRESS OF SECTOR 18
```

```
1067   001544  050360                  .WORD   BUFP+<520.*19.>  ;ADDRESS OF SECTOR 19
1068   001546  051370                  .WORD   BUFP+<520.*20.>  ;ADDRESS OF SECTOR 20
1069   001550  052400                  .WORD   BUFP+<520.*21.>  ;ADDRESS OF SECTOR 21
1070
1071                           ;REMAINING WORD COUNT TABLE
1072
1073   001552  000404          WCTBL:  .WORD   260.             ;REMAINING WORD COUNT AFTER SECTOR 0
1074   001554  001010                  .WORD   260.+<260.*1.>   ;REMAINING WORD COUNT AFTER SECTOR 1
1075   001556  001414                  .WORD   260.+<260.*2.>   ;REMAINING WORD COUNT AFTER SECTOR 2
1076   001560  002020                  .WORD   260.+<260.*3.>   ;REMAINING WORD COUNT AFTER SECTOR 3
1077   001562  002424                  .WORD   260.+<260.*4.>   ;REMAINING WORD COUNT AFTER SECTOR 4
1078   001564  003030                  .WORD   260.+<260.*5.>   ;REMAINING WORD COUNT AFTER SECTOR 5
1079   001566  003434                  .WORD   260.+<260.*6.>   ;REMAINING WORD COUNT AFTER SECTOR 6
1080   001570  004040                  .WORD   260.+<260.*7.>   ;REMAINING WORD COUNT AFTER SECTOR 7
1081   001572  004444                  .WORD   260.+<260.*8.>   ;REMAINING WORD COUNT AFTER SECTOR 8
1082   001574  005050                  .WORD   260.+<260.*9.>   ;REMAINING WORD COUNT AFTER SECTOR 9
1083   001576  005454                  .WORD   260.+<260.*10.>  ;REMAINING WORD COUNT AFTER SECTOR 10
1084   001600  006060                  .WORD   260.+<260.*11.>  ;REMAINING WORD COUNT AFTER SECTOR 11
1085   001602  006464                  .WORD   260.+<260.*12.>  ;REMAINING WORD COUNT AFTER SECTOR 12
1086   001604  007070                  .WORD   260.+<260.*13.>  ;REMAINING WORD COUNT AFTER SECTOR 13
1087   001606  007474                  .WORD   260.+<260.*14.>  ;REMAINING WORD COUNT AFTER SECTOR 14
1088   001610  010100                  .WORD   260.+<260.*15.>  ;REMAINING WORD COUNT AFTER SECTOR 15
1089   001612  010504                  .WORD   260.+<260.*16.>  ;REMAINING WORD COUNT AFTER SECTOR 16
1090   001614  011110                  .WORD   260.+<260.*17.>  ;REMAINING WORD COUNT AFTER SECTOR 17
1091   001616  011514                  .WORD   260.+<260.*18.>  ;REMAINING WORD COUNT AFTER SECTOR 18
1092   001620  012120                  .WORD   260.+<260.*19.>  ;REMAINING WORD COUNT AFTER SECTOR 19
1093   001622  012524                  .WORD   260.+<260.*20.>  ;REMAINING WORD COUNT AFTER SECTOR 20
1094   001624  013130                  .WORD   260.+<260.*21.>  ;REMAINING WORD COUNT AFTER SECTOR 21
1095
1096                           .EVEN
1097
```

```
                        .SBTTL  ERROR POINTER TABLE

                        ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                        ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                        ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                        ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                        ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

                        ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                        ;*      DH              ;;POINTS TO THE DATA HEADER
                        ;*      DT              ;;POINTS TO THE DATA
                        ;*      DF              ;;POINTS TO THE DATA FORMAT

001626                  $ERRTB:
                        ;ERROR 1

001626  022112                 EM1                     ;RH11 INTERRUPT OCCURRED (RPAS=0)
001630  023231                 DH1
001632  024502                 DT1
001634  025014                 DF1

                        ;ERROR 2

001636  022153                 EM2                     ;UNEXPECTED ATTENTION OCCURRED
001640  023236                 DH2
001642  024504                 DT2
001644  025020                 DF2

                        ;ERROR 3

001646  022211                 EM3                     ;MASSBUS PARITY ERROR (MCPE=1)
001650  023313                 DH3
001652  024520                 DT3
001654  025024                 DF3

                        ;ERROR 4

001656  022247                 EM4                     ;MASSBUS PARITY ERROR (PAR=1)
001660  023341                 DH4
001662  024526                 DT4
001664  025030                 DF4

                        ;ERROR 5

001666  022304                 EM5                     ;ADDRESS PLUG CHANGE BIT SET
001670  023236                 DH2
001672  024504                 DT2
001674  025020                 DF2

                        ;ERROR 6

001676  022340                 EM6                     ;RH11 DIDN'T RESPOND TO ADDRESSING
001700  023400                 DH6
001702  024536                 DT6
001704  025014                 DF1
```

```
1154
1155                                    ;ERROR 7
1156
1157   001706  000000                             0                       ;UNUSED
1158   001710  000000                             0
1159   001712  000000                             0
1160   001714  000000                             0
1161
1162                                    ;ERROR 10
1163
1164   001716  022402                             EM10                    ;DRIVE OFFLINE
1165   001720  023413                             DH10
1166   001722  024540                             DT10
1167   001724  025034                             DF10
1168
1169                                    ;ERROR 11
1170
1171   001726  022420                             EM11                    ;PERSISTENT DRIVE UNSAFE ERROR
1172   001730  023413                             DH10
1173   001732  024540                             DT10
1174   001734  025034                             DF10
1175
1176                                    ;ERROR 12
1177
1178   001736  022456                             EM12                    ;UNCORRECTABLE MASSBUS PARITY ERROR
1179   001740  023413                             DH10
1180   001742  024540                             DT10
1181   001744  025034                             DF10
1182
1183                                    ;ERROR 13
1184
1185   001746  022521                             EM13                    ;SOFTWARE TIMEOUT
1186   001750  023413                             DH10
1187   001752  024540                             DT10
1188   001754  025034                             DF10
1189
1190                                    ;ERROR 14
1191
1192   001756  022542                             EM14                    ;DRIVE UNSAFE ERROR
1193   001760  023413                             DH10
1194   001762  024540                             DT10
1195   001764  025034                             DF10
1196
1197                                    ;ERROR 15
1198
1199   001766  022565                             EM15                    ;CONTROLLER/DRIVE ERROR DURING WRITE
1200   001770  023413                             DH10
1201   001772  024540                             DT10
1202   001774  025034                             DF10
1203
1204                                    ;ERROR 16
1205
1206   001776  022631                             EM16                    ;CONTROLLER/DRIVE ERROR DURING WRITE CHECK
1207   002000  023413                             DH10
1208   002002  024540                             DT10
1209   002004  025034                             DF10
```

```
                                ;ERROR 17

002006  022703                          EM17                    ;RETRIES NOT SUCESSFUL - SECTOR NOT ACCEPTABLE
002010  023663                          DH17
002012  024614                          DT17
002014  025050                          DF17

                                ;ERROR 20

002016  022761                          EM20                    ;DATA ERROR DURING WRITE CHECK
002020  023732                          DH20
002022  024626                          DT20
002024  025054                          DF20

                                ;ERROR 21

002026  023017                          EM21                    ;CONTROLLER/DRIVE ERROR VERIFYING HEADERS
002030  023413                          DH10
002032  024540                          DT10
002034  025034                          DF10

                                ;ERROR 22

002036  023070                          EM22                    ;'HCE' ERROR VERIFYING HEADERS
002040  023413                          DH10
002042  024540                          DT10
002044  025034                          DF10

                                ;ERROR 23

002046  023137                          EM23                    ;CYLINDER FIELD IN HEADER IS NOT CORRECT
002050  024231                          DH23
002052  024710                          DT23
002054  025074                          DF23

                                ;ERROR 24

002056  023207                          EM24                    ;WRITE CHECK ERROR
002060  024327                          DH24
002062  024726                          DT24
002064  025100                          DF24

                                ;;*****************************************************************

                                .SBTTL  MAIN PROGRAM

                                ;;*****************************************************************

002066  012737  177777  001302  BEGIN:  MOV     #-1,CHGADR      ;SET CHANGE 'RH11 BUS ADDRESS' INDICATOR
002074  000402                  BEGIN2: BR      BEGIN2          ;START THE PROGRAM
002076  005037  001302          BEGIN1: CLR     CHGADR          ;CLEAR THE 'CHANGE RH11 ADDRESS' INDICATOR
002102  000005                  BEGIN2: RESET                   ;CLEAR THE BUS
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
002104  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
```

```
1266  002110  005026                        CLR     (R6)+                   ;;CLEAR MEMORY LOCATION
1267  002112  022706  001140                CMP     #SWR,R6 ;;DONE?
1268  002116  001374                        BNE     .-6                     ;;LOOP BACK IF NO
1269  002120  012706  001100                MOV     #STACK,SP               ;;SETUP THE STACK POINTER
1270                                ;;INITIALIZE A FEW VECTORS
1271  002124  012737  006636  000030        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1272  002132  012737  000340  000032        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
1273  002140  012737  012014  000034        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1274  002146  012737  000340  000036        MOV     #340,@#TRAPVEC+2;LEVEL 7
1275  002154  005037  001160                CLR     $ESCAPE                 ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1276  002160  112737  000001  001115        MOVB    #1,$ERMAX               ;;ALLOW ONE ERROR PER TEST
1277                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1278                                ;;EQUAL TO A "-1"  SETUP FOR A SOFTWARE SWITCH REGISTER.
1279  002166  013746  000004                MOV     @#ERRVEC,-(SP)          ;;SAVE ERROR VECTOR
1280  002172  012737  002226  000004        MOV     #64$,@#ERRVEC           ;;SET UP ERROR VECTOR
1281  002200  012737  177570  001140        MOV     #DSWR,SWR               ;;SETUP FOR A HARDWARE SWICH REGISTER
1282  002206  012737  177570  001142        MOV     #DDISP,DISPLAY          ;;AND A HARDWARE DISPLAY REGISTER
1283  002214  022777  177777  176716        CMP     #-1,@SWR                ;;TRY TO REFERENCE HARDWARE SWR
1284  002222  001012                        BNE     66$                     ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1285                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
1286  002224  000403                        BR      65$                     ;;BRANCH IF NO TIMEOUT
1287  002226  012716  002234        64$:    MOV     #65$,(SP)               ;;SET UP FOR TRAP RETURN
1288  002232  000002                        RTI
1289  002234  012737  000176  001140  65$:  MOV     #SWREG,SWR              ;;POINT TO SOFTWARE SWR
1290  002242  012737  000174  001142        MOV     #DISPREG,DISPLAY
1291  002250  012637  000004        66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1292
1293  002254  000005                        RESET                           ;CLEAR WORLD
1294  002256  012737  000240  000036        MOV     #PR5,@#TRAPVEC+2  ;CHANGE TRAP PRIORITY BACK TO 5
1295  002264  012737  000240  000032        MOV     #PR5,@#EMTVEC+2 ;CHANGE EMT PRIORITY BACK TO 5
1296  002272  012737  002076  001300        MOV     #BEGIN1,CNTLC   ;'CONTROL C' ADDRESS
```

```
1297   002300  005227  177777          INC     #-1             ;FIRST START ?
1298   002304  001010                  BNE     1$              ;BR IF NOT
1299   002306  104401  025130          TYPE    ,TITLE          ;ADRS OF 'TITLE' MESSAGE
1300   002312  122737  000011  000041  CMPB    #11,41          ;LOADED FROM AN RP04/5/6 ?
1301   002320  001002                  BNE     1$              ;BR IF NOT
1302   002322  104401  025205          TYPE    ,LOADRV         ;INSTRUCT OPERATOR TO REMOVE 'XXDP'
1303                                                          ;PACK FROM DRIVE 0
1304   002326  004737  010204  1$:     JSR     PC,$TKINT       ;TURN ON THE KEYBOARD INTERRUPT
1305                          .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1306   002332  005737  000042          TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
1307   002336  001006                  BNE     67$             ;;BRANCH IF YES
1308   002340  023727  001140  000176  CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
1309   002346  001005                  BNE     68$             ;;BRANCH IF NO
1310   002350  104406                  GTSWR                   ;;GET SOFT-SWR SETTINGS
1311   002352  000403                  BR      68$
1312   002354  112737  000001  001134  67$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
1313   002362                          68$:
1314   002362  005227  177777          INC     #-1             ;CHECK FOR FIRST START
1315   002366  001010                  BNE     SETVEC          ;BR IF NOT
1316   002370  004737  025432          JSR     PC,BUSADR       ;CHECK THE RH11 ADDRESS
1317   002374  013737  001172  012246  MOV     $RPADR,RPADR    ;RH11 ADDRESS
1318   002402  013737  001174  012250  MOV     $RPVEC,RPVEC    ;RH11 VECTOR ADDRESS
1319
1320                          ;DISPLAY DRIVE STATUS AND SET UP THE OTHER UNITS THE
1321                          ;       PROGRAM WILL USE
1322
1323   002410  004737  005742  SETVEC: JSR     PC,ST.CLK       ;START THE CLOCK
1324   002414  004737  012264          JSR     PC,RPINIT       ;INITIALIZE THE RP04/5/6 DRIVER
1325   002420  012737  177777  012206  MOV     #-1,SAVEFG      ;SET THE SAVE REGISTERS FLAG
1326   002426  005227  177777          INC     #-1             ;SEE IF FIRST START
1327   002432  001404                  BEQ     11$             ;BR IF YES
1328   002434  032777  000004  176476  BIT     #SW02,@$SWR     ;TYPEOUT THE DRIVE STATUS TABLE ?
1329   002442  001076                  BNE     10$             ;BR IF NOT
1330   002444  012737  000340  177776  11$:    MOV     #PR7,PS         ;SET PRIORITY TO 7
1331   002452  005004                  CLR     R4              ;DRIVE TABLE POINTER
1332   002454  104401  001167          TYPE    ,$CRLF          ;CR-LF
1333   002460  104401  020640          TYPE    ,SYSTAT         ;TYPE STATUS HEADING
1334   002464                          1$:
1335   002464  010446                  MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
1336                                                          ;;TYPE DRIVE NUMBER
1337   002466  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
1338   002470     002                  .BYTE   2               ;;TYPE 2 DIGIT(S)
1339   002471     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
1340   002472  104401  020703          TYPE    ,LIN4SP         ;SPACES
1341   002476  105764  012120          TSTB    DRVSTA(R4)      ;CHECK DRIVE'S STATUS
1342   002502  100416                  BMI     4$              ;BR IF UNSAFE
1343   002504  001020                  BNE     5$              ;BR IF ONLINE
1344   002506  105764  012130          TSTB    DRVTYP(R4)      ;SEE IF OFFLINE OR NONEXISTENT
1345   002512  001404                  BEQ     2$              ;BR IF NONEXISTENT
1346   002514  100006                  BPL     3$              ;BR IF OFFLINE
1347   002516  104401  020553          TYPE    ,NOTRP          ;DRIVE NOT AN RP04/5/6
1348   002522  000440                  BR      9$              ;CHECK NEXT DRIVE
1349   002524  104401  020574  2$:     TYPE    ,NOTPRS         ;DRIVE NOT PRESENT
1350   002530  000435                  BR      9$              ;CHECK NEXT DRIVE
1351   002532  104401  020532  3$:     TYPE    ,UNTOFF         ;DRIVE OFFLINE
1352   002536  000405                  BR      6$              ;PRINT DRIVE TYPE
```

```
1353  002540  104401  020611          4$:     TYPE    ,NOTSAF         ;DRIVE UNSAFE
1354  002544  000402                  BR      6$              ;PRINT DRIVE TYPE
1355  002546  104401  020543          5$:     TYPE    ,UNTON          ;DRIVE ONLINE
1356  002552  104401  020705          6$:     TYPE    ,LINSP          ;SPACES
1357  002556  012737  020621  002622          MOV     #RP04B,8$       ;ADDRESS OF RP04 MESSAGE
1358  002564  132764  000001  012130          BITB    #BIT00,DRVTYP(R4)   ;RP04 ?
1359  002572  001012                          BNE     7$              ;BR IF YES
1360  002574  012737  020626  002622          MOV     #RP05,8$        ;ADDRESS OF RP05 MESSAGE
1361  002602  132764  000002  012130          BITB    #BIT01,DRVTYP(R4)   ;RP05 ?
1362  002610  001003                          BNE     7$              ;BR IF YES
1363  002612  012737  020633  002622          MOV     #RP06,8$        ;ADDRESS OF RP06 MESSAGE
1364  002620  104401                  7$:     TYPE                    ;TYPE THE DRIVE TYPE MESSAGE
1365  002622  000000                  8$:     .WORD   0               ;MESSAGE ADDRESS HERE
1366  002624  104401  001167          9$:     TYPE    ,$CRLF          ;CR-LF
1367  002630  005204                          INC     R4              ;INCREMENT DRIVE NUMBER/TABLE POINTER
1368  002632  020427  000010                  CMP     R4,#8.          ;FINISHED ?
1369  002636  001312                          BNE     1$              ;BR IF NOT
1370  002640  104401  001167          10$:    TYPE    ,$CRLF          ;CR-LF
1371
1372
1373                                  ;SEE WHICH MODE THE PROGRAM IS TO BE RUN IN.
1374                                  ;MODES ARE: 'FORMAT & VERIFY' OR 'CHECK FORMAT'
1375
1376  002644  005037  001220          M1:     CLR     MODE            ;SET MODE TO 'CHECK FORMAT'
1377  002650  104401  021121                  TYPE    ,MMODE          ;TYPE 'PROGRAM MODE'
1378  002654  104411                          RDLIN                   ;READ THE KEYBOARD
1379  002656  012601                          MOV     (SP)+,R1        ;GET ADDRESS OF INPUT
1380  002660  105711                          TSTB    (R1)            ;'CR' ENTERED (DEFAULT)
1381  002662  001414                          BEQ     2$              ;BR IF YES
1382  002664  122711  000103                  CMPB    #'C,(R1)        ;'CHECK' ?
1383  002670  001406                          BEQ     1$              ;BR IF YES
1384  002672  122711  000106                  CMPB    #'F,(R1)        ;'FORMAT' ?
1385  002676  001411                          BEQ     3$              ;BR IF YES
1386  002700  104401  001166                  TYPE    ,$QUES          ;NO CORRECT ENTRY
1387  002704  000757                          BR      M1              ;TRY AGAIN
1388  002706  104401  021174          1$:     TYPE    ,MHECK          ;TYPE REST OF 'CHECK'
1389  002712  000410                          BR      M1A             ;GET STARTING ADDRESS
1390  002714  104401  021153          2$:     TYPE    ,MFORMT         ;TYPE DEFAULT MESSAGE
1391  002720  000402                          BR      4$              ;SET UP MODE
1392  002722  104401  021207          3$:     TYPE    ,MORMAT         ;TYPE REST OF 'FORMAT'
1393  002726  012737  177777  001220  4$:     MOV     #-1,MODE        ;SET MODE TO 'FORMAT & VERIFY'
1394
1395                                  ;FIND OUT IF FORMAT IS TO BE IN 20 OR 22 SECTOR MODE
1396
1397  002734  104401  021227          M1A:    TYPE    ,MSIZE          ;TYPE FORMAT MODE REQUEST
1398  002740  104411                          RDLIN                   ;READ THE KEYBOARD
1399  002742  012601                          MOV     (SP)+,R1        ;ADDRESS OF ENTRY
1400  002744  122711  000131                  CMPB    #'Y,(R1)        ;IS ENTRY A 'Y' ?
1401  002750  001410                          BEQ     1$              ;BR IF IT IS
1402  002752  122711  000116                  CMPB    #'N,(R1)        ;IS ENTRY A 'N' ?
1403  002756  001424                          BEQ     2$              ;BR IF IT IS
1404  002760  105711                          TSTB    (R1)            ;DEFAULT MODE ?
1405  002762  001403                          BEQ     1$              ;BR IF IT IS
1406  002764  104401  001166                  TYPE    ,$QUES          ;TYPE '?'
1407  002770  000761                          BR      M1A             ;TRY AGAIN
1408  002772  104401  021300          1$:     TYPE    ,MSEC22         ;TYPEOUT MODE SELECTED
```

GET VALUE FOR SOFTWARE SWITCH REGISTER SEQ 0028

```
1409  002776  012737  013130  001256          MOV     #<260.*22.>,WC  ;TRACK SIZE IN 22 SECTOR MODE
1410  003004  012737  164650  001260          MOV     #-<260.*22.>,MWC  ;2'S COMPLEMENT WORD COUNT
1411  003012  012737  177777  001264          MOV     #-1,SEC20       ;22 SECTOR INDICATOR
1412  003020  012737  000025  001266          MOV     #21.,MAXSEC     ;MAX SECTOR ADDRESS IN 22 SECTOR MODE
1413  003026  000415                          BR      M1B             ;CONTINUE
1414  003030  104401  021357          2$:     TYPE    ,MSEC20         ;TYPE OUT 20 SECTOR MODE SELECTED
1415  003034  012737  012120  001256          MOV     #<260.*20.>,WC  ;TRACK SIZE IN 20 SECTOR MODE
1416  003042  012737  165660  001260          MOV     #-<260.*20.>,MWC  ;2'S COMPLEMENT WORD COUNT
1417  003050  005037  001264                  CLR     SEC20           ;20 SECTOR INDICATOR
1418  003054  012737  000023  001266          MOV     #19.,MAXSEC     ;MAX SECTOR ADDRESS IN 20 SECTOR MODE
1419  003062  005037  001230          M1B:    CLR     BEGTRK          ;CLEAR STARTING TRACK ADDRESS
1420  003066  005037  001224                  CLR     BEGCYL          ;CLEAR BEGINNING CYLINDER ADDRESS
1421  003072  012737  000022  001226          MOV     #18.,ENDTRK     ;SETUP END TRACK ADDRESS
1422  003100  012737  000002  001240          MOV     #2,PATSEL       ;SETUP FOR WORST CASE PATTERN
1423  003106  112737  177777  001354          MOVB    #-1,FMTDPB      ;SETUP INVALID DRIVE NUMBER
1424  003114  000400                          BR      M0              ;BRANCH TO START
1425
1426                                  ;GO FIND OUT WHAT DRIVE
1427
1428  003116  012737  006116  001300  M0:     MOV     #0ENTER,CNTLC   ;CONTROL C ABORT ENTRANCE
1429  003124  005037  001112                  CLR     $ERTTL          ;CLEAR THE ERROR ACCUMULATOR
1430  003130  004737  010204                  JSR     PC,$TKINT       ;INITIALIZE THE TTY KEYBOARD
1431  003134  104401  020657                  TYPE    ,MUNIT          ;ASK FOR DRIVE NUMBER
1432  003140  104411                          RDLIN                   ;READ THE KEYBOARD
1433  003142  012601                          MOV     (SP)+,R1        ;ADDRESS OF ENTRY
1434  003144  004537  006426                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
1435  003150  003176                          3$                      ;ILLEGAL CHARACTER
1436  003152  003164                          2$                      ;CARRIAGE RETURN
1437  003154  003176                          3$                      ;" "
1438  003156  003164                          2$                      ;","
1439  003160  003204                          4$                      ;DIGIT 0-7
1440  003162  003176                          3$                      ;DIGIT 8-9
1441  003164  005037  001214          2$:     CLR     DRIVE           ;SELECT DRIVE ZERO
1442  003170  104401  020701                  TYPE    ,MDRVD          ;GO TYPE DEFAULT DRIVE NUMBER
1443  003174  000413                          BR      5$              ;GO SEE IF DRIVE ZERO IS THERE
1444  003176  104401  001166          3$:     TYPE    ,$QUES          ;TYPE '?'
1445  003202  000745                          BR      M0              ;ASK FOR DRIVE NUMBER AGAIN
1446  003204  010237  001214          4$:     MOV     R2,DRIVE        ;SAVE DRIVE NUMBER
1447  003210  005702                          TST     R2              ;SEE IF DRIVE 0
1448  003212  001004                          BNE     5$              ;BR IF NOT
1449  003214  122737  000011  000041          CMPB    #11,41          ;PROGRAM LOADED FROM AN RP04/5/6 ?
1450  003222  001431                          BEQ     7$              ;BR IF IT WAS, CAN'T FORMAT THE DRIVE
1451  003224  004737  005742          5$:     JSR     PC,ST.CLK       ;START THE CLOCK
1452  003230  004737  012264                  JSR     PC,RPINIT       ;GO SEE WHAT DRIVES ARE AVAILABLE
1453  003234  012737  177777  012206          MOV     #-1,SAVEFG      ;SAVE THE REGISTERS
1454  003242  012737  177777  012210          MOV     #-1,SEEKFG      ;SET 'NO OPTIMIZATION' FLAG
1455  003250  005037  177776                  CLR     PS              ;SET PRIORITY BACK TO ZERO
1456  003254  105762  012120                  TSTB    DRVSTA(R2)      ;LOOK AT DRIVE STATUS
1457  003260  003015                          BGT     8$              ;BRANCH IF ONLINE
1458  003262  001403                          BEQ     6$              ;BR IF DRIVE NOT AVAILABLE
1459  003264  104401  021067                  TYPE    ,MUSDR          ;'DRIVE UNSAFE'
1460  003270  000712                          BR      M0              ;GO GET DRIVE NUMBER AGAIN
1461  003272  105762  012130          6$:     TSTB    DRVTYP(R2)      ;A DRIVE PRESENT?
1462  003276  001003                          BNE     7$              ;BR IF SO
1463  003300  104401  020762                  TYPE    ,MDRNP          ;TYPE 'DRIVE NOT PRESENT'
1464  003304  000704                          BR      M0              ;GO GET DRIVE NUMBER AGAIN
```

```
1465  003306  104401  021007          7$:     TYPE    ,MER11          ;'DRIVE NOT AVAILABLE'
1466  003312  000701                          BR      M0              ;GO GET DRIVE NUMBER AGAIN
1467  003314  123737  001214  001354  8$:     CMPB    DRIVE,FMTDPB    ;SAME DRIVE AS LAST TIME ?
1468  003322  001430                          BEQ     M2              ;BR IF IT IS
1469  003324  113737  001214  001354          MOVB    DRIVE,FMTDPB    ;SETUP DRIVE ADDRESS
1470  003332  012737  000632  001222          MOV     #410.,ENDCYL    ;SETUP FOR RP04/5
1471  003340  132762  000003  012130          BITB    #BIT00!BIT01,DRVTYP(R2)  ;SEE IF DRIVE RP04/5
1472  003346  001003                          BNE     9$              ;BR IF EITHER
1473  003350  012737  001456  001222          MOV     #814.,ENDCYL    ;SETUP ENDING CYLINDER FOR RP06
1474  003356  005037  001230          9$:     CLR     BEGTRK          ;CLEAR STARTING TRACK ADDRESS
1475  003362  005037  001224                  CLR     BEGCYL          ;CLEAR STARTING CYLINDER ADDRESS
1476  003366  013737  001222  001376          MOV     ENDCYL,TABLE+2  ;ENTRY LIMIT FOR BEGINNING CYLINDER
1477  003374  013737  001222  001412          MOV     ENDCYL,TABLE+16 ;ENTRY LIMIT FOR END CYLINDER
1478  003402  000400                          BR      M2              ;GET ADDRESS LIMITS FROM THE OPERATOR
1479
1480                                  ;GET ADDRESS LIMITS
1481
1482  003404  104401  020710          M2:     TYPE    ,ENTADR         ;'ENTER ADDRESS LIMITS'
1483  003410  004737  006236                  JSR     PC,PARENT       ;GET THE ADDRESS LIMITS
1484  003414  023737  001222  001224          CMP     ENDCYL,BEGCYL   ;SEE IF ENDING CYLINDER EQ TO GT THAN BEGINNING
1485  003422  101010                          BHI     M4              ;BR IF HIGHER
1486  003424  103404                          BLO     3$              ;BR IF LESS
1487  003426  023737  001226  001230  2$:     CMP     ENDTRK,BEGTRK   ;SEE IF ENDING TRACK EQ OR GT THAN BEGINNING
1488  003434  103003                          BHIS    M4              ;BR IF YES
1489  003436  104401  021456          3$:     TYPE    ,MADRER         ;INVALID ADDRESS ENTERED
1490  003442  000760                          BR      M2              ;TRY AGAIN
1491
1492                                  ;GO GET DATA PATTERN FOR FORMAT
1493
1494  003444  104401  021560          M4:     TYPE    ,MSELP          ;GO TYPE 'SELECT PATTERN'
1495  003450  104411                          RDLIN                   ;READ THE KEYBOARD
1496  003452  012601                          MOV     (SP)+,R1        ;ENTRY ADDRESS
1497  003454  004537  006426                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
1498  003460  003514                          3$                      ;ILLEGAL CHARACTER
1499  003462  003474                          1$                      ;CARRIAGE RETURN
1500  003464  003514                          3$                      ;" "
1501  003466  003474                          1$                      ;"'"
1502  003470  003506                          2$                      ;DIGIT 0-7
1503  003472  003514                          3$                      ;DIGIT 8-9
1504  003474  104401  021660          1$:     TYPE    ,MPATD          ;TYPE DEFAULT PATTERN
1505  003500  012702  000002                  MOV     #2,R2           ;WORST CASE PATTERN
1506  003504  000406                          BR      4$              ;GO SAVE PATTERN
1507  003506  020227  000002          2$:     CMP     R2,#2           ;IS # LARGER THAN 2
1508  003512  101403                          BLOS    4$              ;BRANCH IF NOT
1509  003514  104401  001166          3$:     TYPE    ,$QUES          ;TYPE '?'
1510  003520  000751                          BR      M4              ;RETYPE LINE
1511  003522  010237  001240          4$:     MOV     R2,PATSEL       ;SAVE PATTERN SELECTED
1512
1513                                  ;GO TYPE 'STARTING FORMAT ON DRIVE N'
1514
1515  003526  012737  006136  001300  M5:     MOV     #TYPADR,CNTLC   ;CHANGE ^C ENTRANCE
1516  003534  005737  001220                  TST     MODE            ;'FORMAT' OR 'CHECK' MODE ?
1517  003540  001403                          BEQ     1$              ;BR IF 'CHECK' MODE
1518  003542  104401  021673                  TYPE    ,MSFOU          ;TYPE 'STARTING FORMAT ON DRIVE N'
1519  003546  000402                          BR      2$
1520  003550  104401  021730          1$:     TYPE    ,MSCHK          ;TYPE 'STARTING CHECK ON DRIVE N'
```

GET VALUE FOR SOFTWARE SWITCH REGISTER SEQ 0030

```
1521   003554                          2$:
1522   003554  013746  001214                  MOV     DRIVE,-(SP)     ;;SAVE DRIVE FOR TYPEOUT
1523                                                                   ;;TYPE DRIVE NUMBER
1524   003560  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
1525   003562     001                          .BYTE   1               ;;TYPE 1 DIGIT(S)
1526   003563     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
1527   003564  104401  001167                  TYPE    ,$CRLF          ;CR-LF
1528
1529                                  ;SETUP TOTAL TRACK COUNT FOR FORMAT
1530
1531   003570  023737  001222  001224  CKADRS: CMP     ENDCYL,BEGCYL   ;STARTING AND ENDING CYLINDERS THE SAME ?
1532   003576  001011                          BNE     1$              ;BRANCH IF BEGCYL NOT EQUAL TO ENDCYL
1533   003600  013737  001226  001232          MOV     ENDTRK,TTRKS    ;END TRACK ADDRESS
1534   003606  163737  001230  001232          SUB     BEGTRK,TTRKS    ;SUBTRACT THE STARTING TRACK ADDRESS
1535   003614  005237  001232                  INC     TTRKS           ;MAKE THE NUMBER OF TRACKS INCLUSIVE
1536   003620  000427                          BR      SETPAT          ;SETUP THE DATA PATTERN
1537   003622  013702  001222          1$:     MOV     ENDCYL,R2       ;ENDING CYLINDER
1538   003626  163702  001224                  SUB     BEGCYL,R2       ;SUBTRACT THE STARTING CYLINDER
1539   003632  013737  001226  001232          MOV     ENDTRK,TTRKS    ;CALCULATE THE RESIDUAL TRACKS
1540   003640  163737  001230  001232          SUB     BEGTRK,TTRKS    ;SUBTRACT THE STARTING TRACK
1541   003646  100004                          BPL     2$              ;BR IF ENDING TRACK GREATER
1542   003650  062737  000023  001232          ADD     #19.,TTRKS      ;CORRECT THE VALUE
1543   003656  005302                          DEC     R2              ;COUNT THE PARTIAL TRACK
1544   003660  005237  001232          2$:     INC     TTRKS           ;MAKE THE NUMBER INCLUSIVE
1545   003664  005302                  3$:     DEC     R2              ;DECREMENT THE CYLINDER COUNT
1546   003666  100404                          BMI     SETPAT          ;BR IF DONE
1547   003670  062737  000023  001232          ADD     #19.,TTRKS      ;ADD CYLINDER WORTH OF TRACKS
1548   003676  000772                          BR      3$              ;CONTINUE
1549
1550                                  ;THIS CODE SETS UP THE SELECTED DATA PATTERN TO BE WRITTEN ON EACH SECTOR IMAGE
1551
1552   003700  005037  001242  SETPAT: CLR     PATA            ;CLEAR DATA PATTERN A
1553   003704  005037  001244          CLR     PATB            ;CLEAR DATA PATTERN B
1554   003710  005737  001240          TST     PATSEL          ;SEE IF PATTERN OF ONES
1555   003714  001416                  BEQ     1$              ;BR IF SO
1556   003716  005137  001242          COM     PATA            ;SET PATA TO ONES
1557   003722  005137  001244          COM     PATB            ;SET PATB TO ONES
1558   003726  022737  000001  001240  CMP     #1,PATSEL       ;SEE IF PATTERN OF ZEROS
1559   003734  001406                  BEQ     1$              ;BRANCH IF SO
1560   003736  012737  165555  001242  MOV     #165555,PATA    ;SET UP WORST CASE
1561   003744  012737  133333  001244  MOV     #133333,PATB    ;SET UP WORST CASE
1562   003752  013703  001256  1$:     MOV     WC,R3           ;SET UP COUNTER
1563   003756  012700  025130          MOV     #BUFP,R0        ;SET UP MEMORY POINTER
1564   003762  013701  001242          MOV     PATA,R1         ;SET UP PATTERN IN R1
1565   003766  013702  001244          MOV     PATB,R2         ;SET UP PATTERN IN R2
1566   003772  010120          2$:     MOV     R1,(R0)+        ;MOV 1ST PAT INTO MEM
1567   003774  010220                  MOV     R2,(R0)+        ;MOV 2ND PAT INTO MEM
1568   003776  005303                  DEC     R3              ;KEEP COUNT
1569   004000  005303                  DEC     R3              ;KEEP COUNT
1570   004002  001373                  BNE     2$              ;DO IT AGAIN IF R3 NOT 0
1571
1572                                  ;THIS CODE SETS UP FOR THE ACTUAL FORMAT
1573
1574   004004  004737  005402  WRTRK:  JSR     PC,SETTBL       ;GO SET UP DRIVER TABLE
1575   004010  004737  005602          JSR     PC,SETHDR       ;GO INITIALIZE HEADERS IN THE SECTOR BUFFER IN CORE
1576   004014  112737  000020  001355  MOVB    #20,FMTDPB+1    ;LOAD FMT22 BIT
```

```
1577   004022   005737   001264                    TST     SEC20           ;18 BIT MODE ?
1578   004026   001002                             BNE     1$              ;BR IF NOT
1579   004030   105037   001355                    CLRB    FMTDPB+1        ;CLEAR THE FMT22 BIT
1580   004034   112737   000143   001356  1$:      MOVB    #SETFMT,FMTDPB+2 ;'LOAD FORMAT' COMMAND
1581   004042   004037   013034           2$:      JSR     R0,RP04         ;START THE COMMAND
1582   004046   001354                             FMTDPB                  ;DPB ADDRESS
1583   004050   000774                             BR      2$              ;QUEUE FULL RETURN
1584   004052   005737   001372           3$:      TST     FMTDPB+16       ;LOOK FOR DONE
1585   004056   001775                             BEQ     3$              ;LOOP UNTIL FINISHED
1586   004060   005037   001216           WRTRK1:  CLR     $OFSW           ;CLEAR ERROR COUNTER
1587   004064   005037   001250                    CLR     RETRY           ;ZERO THE RETRY COUNTER
1588   004070   105037   001364                    CLRB    FMTDPB+10       ;RESTORE SECTOR
1589   004074   013737   001260   001360           MOV     MWC,FMTDPB+4    ;RESTORE WC
1590   004102   012737   025130   001362           MOV     #BUFP,FMTDPB+6  ;RESTORE BA
1591   004110   005737   001220                    TST     MODE            ;'FORMAT' OR 'CHECK' MODE ?
1592   004114   001450                             BEQ     CKTRK           ;BR IF 'CHECK' MODE
1593   004116   112737   000163   001356  WRTRK2:  MOVB    #WRTHD,FMTDPB+2 ;SET WRITE HEADER & DATA COMMAND IN TBL
1594   004124   012737   004124   001110           MOV     #.,$LPERR       ;SETUP LOOP ON ERROR ADDRESS
1595   004132   012706   001100                    MOV     #$STACK,SP      ;LOAD STACK POINTER
1596   004136   004037   013034           1$:      JSR     R0,RP04         ;GO FORMAT A TRACK
1597   004142   001354                             FMTDPB                  ;ADRS OF PARAMETERS - TBL
1598   004144   000774                             BR      1$              ;WAIT FOR QUEUE IF FULL
1599   004146   005737   001372           2$:      TST     FMTDPB+16       ;WAIT FOR COMMAND TO COMPLETE
1600   004152   001775                             BEQ     2$              ;BRANCH IF NOT DONE
1601   004154   100024                             BPL     4$              ;BRANCH IF NO ERROR
1602   004156   012737   004204   001160           MOV     #3$,$ESCAPE     ;ESCAPE TO 3$ ON ERROR
1603   004164   004737   005234                    JSR     PC,ERINDX       ;SEE WHICH ERROR
1604   004170   104010                             ERROR   10              ;DRIVE OFFLINE
1605   004172   104011                             ERROR   11              ;PERSISTENT DRIVE UNSAFE ERROR
1606   004174   104012                             ERROR   12              ;UNCORRECTABLE MASSBUS PARITY ERROR
1607   004176   104013                             ERROR   13              ;SOFTWARE TIMEOUT
1608   004200   104014                             ERROR   14              ;DRIVE UNSAFE ERROR
1609   004202   104015                             ERROR   15              ;DRIVE/CONTROLLER ERROR DURING WRITE
1610   004204   004737   005334           3$:      JSR     PC,LOP.CK       ;LOOP ON THE ERROR ?
1611   004210   022737   000003   001250           CMP     #3,RETRY        ;ERROR RETRY LIMIT ?
1612   004216   001403                             BEQ     4$              ;BR IF REACHED
1613   004220   005237   001250                    INC     RETRY           ;COUNT THE ERROR
1614   004224   000744                             BR      1$              ;TRY AGAIN
1615   004226   005037   001160           4$:      CLR     $ESCAPE         ;CLEAR ERROR ESCAPE ADDRESS
1616   004232   005037   001250                    CLR     RETRY           ;CLEAR THE RETRY COUNTER
1617
1618                                     ;CHECK THE TRACK JUST WRITTEN
1619
1620   004236   112737   000153   001356  CKTRK:   MOVB    #WCKHD,FMTDPB+2 ;SET WRITE CHECK HEADER & DATA COMMAND IN TBL
1621   004244   012737   004244   001110           MOV     #.,$LPERR       ;SETUP LOOP ON ERROR ADDRESS
1622   004252   012706   001100                    MOV     #$STACK,SP      ;LOAD STACK POINTER
1623   004256   004037   013034           1$:      JSR     R0,RP04         ;GO CHECK THE TRACK JUST FORMATTED
1624   004262   001354                             FMTDPB                  ;ADRS OF PARAMETERS - TBL
1625   004264   000774                             BR      1$              ;WAIT FOR QUEUE IF FULL
1626   004266   005737   001372           2$:      TST     FMTDPB+16       ;WAIT FOR COMMAND TO COMPLETE
1627   004272   001775                             BEQ     2$              ;BRANCH IF NOT DONE
1628   004274   100112                             BPL     8$              ;BRANCH IF NO ERROR
1629   004276   113737   001312   001252           MOVB    RP.REG+RPDA,SAVSEC  ;GET THE SECTOR ADDRESS
1630   004304   001005                             BNE     3$              ;BRANCH IF NOT SECTOR 0
1631   004306   013737   001266   001252           MOV     MAXSEC,SAVSEC   ;RESTORE TO LAST SECTOR +1
1632   004314   005237   001252                    INC     SAVSEC          ;INCREMENT TO LAST SECTOR +1
```

```
1633  004320  005337  001252          3$:     DEC     SAVSEC          ;ADJUST SECTOR TO THE ONE THAT FAILED
1634  004324  012737  004570  001160          MOV     #10$,$ESCAPE    ;;ESCAPE TO 10$ ON ERROR
1635  004332  004737  005234                  JSR     PC,ERINDX       ;SEE WHICH ERROR
1636  004336  104010                          ERROR   10              ;DRIVE OFFLINE
1637  004340  104011                          ERROR   11              ;PERSISTENT DRIVE UNSAFE ERROR
1638  004342  104012                          ERROR   12              ;UNCORRECTABLE MASSBUS PARITY ERROR
1639  004344  104013                          ERROR   13              ;SOFTWARE TIMEOUT
1640  004346  104014                          ERROR   14              ;DRIVE UNSAFE ERROR
1641  004350  032737  040000  001314          BIT     #WCE,RP.REG+RPCS2  ;WRITE CHECK ERROR ?
1642  004356  001005                          BNE     4$              ;BR IF SET
1643  004360  033727  001320                  BIT     RP.REG+RPER1,(PC)+  ;CHECK FOR DATA ERRORS
1644  004364  130620                          .WORD   DCK!OPI!DTE!NCRC!HCE!FER  ;DATA ERROR BITS
1645  004366  001001                          BNE     4$              ;BR IF SET
1646  004370  104016                          ERROR   16              ;CONTROLLER/DRIVE ERROR DURING WRITE CHECK
1647  004372  005737  001220          4$:     TST     MODE            ;FORMAT OR CHECK MODE ?
1648  004376  001407                          BEQ     5$              ;BR IF CHECK
1649  004400  005737  001216                  TST     SOFSW           ;RETRYING THE SECTOR ?
1650  004404  001404                          BEQ     5$              ;BR IF NOT
1651  004406  012737  004612  001160          MOV     #11$,$ESCAPE    ;;ESCAPE TO 11$ ON ERROR
1652  004414  104017                          ERROR   17              ;SECTOR NOT ACCEPTABLE
1653  004416  005037  001160          5$:     CLR     $ESCAPE         ;CLEAR THE ERROR ESCAPE ADDRESS
1654  004422  032737  040000  001316          BIT     #ERR,RP.REG+RPDS1  ;DATA ERROR ?
1655  004430  001002                          BNE     6$              ;BR IF IT IS
1656  004432  104024                          ERROR   24              ;WRITE CHECK ERROR
1657  004434  000401                          BR      7$              ;CONTINUE
1658  004436  104020                  6$:     ERROR   20              ;DATA ERROR DURING WRITE CHECK
1659  004440  013701  001252          7$:     MOV     SAVSEC,R1       ;FAILING SECTOR ADDRESS
1660  004444  113737  001252  001364          MOVB    SAVSEC,FMTDPB+10  ;SECTOR ADDRESS
1661  004452  006301                          ASL     R1              ;SETUP INDEX TO ADDRESS WORDS
1662  004454  016137  001476  001362          MOV     ADRTBL(R1),FMTDPB+6  ;BUFFER ADDRESS FOR FAILING SECTOR
1663  004462  013746  001260                  MOV     MWC,-(SP)       ;GET MAXIMUM WORD COUNT VALUE (2'S COMP)
1664  004466  066116  001552                  ADD     WCTBL(R1),(SP)  ;ADD WORD COUNT THROUGH FAILING SECTOR
1665  004472  012637  001254                  MOV     (SP)+,SAVWC     ;STORE WORD COUNT FOR REMAINDER OF TRACK
1666  004476  005737  001220                  TST     MODE            ;FORMAT OR CHECK MODE ?
1667  004502  001421                          BEQ     9$              ;BR IF CHECK
1668  004504  012737  177374  001360          MOV     #-260.,FMTDPB+4 ;WORD COUNT FOR 1 SECTOR
1669  004512  005237  001216                  INC     SOFSW           ;INDICATE THAT A RETRY IS IN PROGRESS
1670  004516  000137  004116                  JMP     WRTRK2          ;REFORMAT ERROR SECTOR
1671  004522  005737  001216          8$:     TST     SOFSW           ;RETRY IN PROGRESS ?
1672  004526  001431                          BEQ     11$             ;BR IF NOT
1673  004530  022737  000002  001216          CMP     #2,SOFSW        ;SEE IF LAST RETRY
1674  004536  001403                          BEQ     9$              ;BR IF IT IS
1675  004540  005237  001216                  INC     SOFSW           ;INCREMENT RETRY COUNT
1676  004544  000644                          BR      1$              ;READ AGAIN
1677  004546  123737  001266  001364  9$:     CMPB    MAXSEC,FMTDPB+10  ;SEE IF LAST SECTOR ON THE TRACK
1678  004554  001416                          BEQ     11$             ;BR IF IT IS
1679  004556  004737  005554                  JSR     PC,SCAWC        ;SETUP TO CHECK REMAINING SECTORS ON THE TRACK
1680  004562  005037  001216                  CLR     SOFSW           ;CLEAR RETRY COUNTER
1681  004566  000633                          BR      1$              ;FINISH CHECKING THE TRACK
1682  004570  004737  005334          10$:    JSR     PC,LOP.CK       ;CHECK FOR LOOP ON ERROR
1683  004574  022737  000003  001250          CMP     #3,RETRY        ;ERROR RETRY REACHED ?
1684  004602  001403                          BEQ     11$             ;BR IF IT IS
1685  004604  005237  001250                  INC     RETRY           ;COUNT THE RETRY
1686  004610  000622                          BR      1$              ;DO THE WRITE CHECK AGAIN
1687  004612  005037  001160          11$:    CLR     $ESCAPE         ;CLEAR THE ERROR RETURN ESCAPE ADDRESS
1688  004616  005037  001250                  CLR     RETRY           ;CLEAR THE RETRY COUNTER
```

GET VALUE FOR SOFTWARE SWITCH REGISTER SEQ 0033

```
 1689  004622  032777  000002  174310          BIT     #BIT1,@SWR      ;LOOP ON CURRENT TRACK ?
 1690  004630  001402                          BEQ     12$             ;BR IF NOT
 1691  004632  000137  004060                  JMP     WRTRK1          ;START AGAIN ON THE SAME TRACK
 1692  004636  004737  005472          12$:    JSR     PC,TRKTST       ;GET UPDATED TRACK AND CYLINDER ADDRESSES
 1693  004642  000402                          BR      HDREAD          ;RETURN HERE IF DONE - DO QUICK CHECK OF DISK
 1694  004644  000137  004060                  JMP     WRTRK1          ;CONTINUE WITH THE FORMAT
 1695
 1696                                  ;THIS CODE MAKES SURE EACH CYLINDER FORMATTED CONTAINS THE
 1697                                  ;PROPER CYLINDER ADDRESS. THE PROGRAM IS LOOKING FOR
 1698                                  ;POSSIBLE POSITIONER ERRORS THAT MAY HAVE OCCURRED DURING THE FORMAT.
 1699
 1700  004650  005037  001262          HDREAD: CLR     HEDERR          ;CLEAR HEADER CHECK ERROR INDICATOR
 1701  004654  004737  005402                  JSR     PC,SETTBL       ;GO SET UP DRIVER TABLE
 1702  004660  004737  005602                  JSR     PC,SETHDR       ;GO SET UP HEADERS IN CORE
 1703  004664  013737  001224  001246          MOV     BEGCYL,CYLCK    ;USE 'BEGCYL' FOR FORMAT VERIFICATION
 1704  004672  012737  177774  001360          MOV     #-4,FMTDPB+4    ;SET UP WORD COUNT
 1705  004700  012737  025120  001362          MOV     #RBUF,FMTDPB+6  ;SET UP BUFFER ADRS
 1706  004706  005037  001364                  CLR     FMTDPB+10       ;CLEAR THE SECTOR & TRACK ADDRESS FIELD
 1707  004712  013737  001224  001366          MOV     BEGCYL,FMTDPB+12  ;SETUP THE CYLINDER FIELD
 1708  004720  112737  000173  001356          MOVB    #RDHD,FMTDPB+2  ;SET UP READ HEADER & DATA COMMAND
 1709  004726  012737  004726  001110          MOV     #.,$LPERR       ;SETUP LOOP ON ERROR ADDRESS
 1710  004734  012706  001100                  MOV     #STACK,SP       ;LOAD STACK POINTER
 1711  004740  004037  013034          1$:     JSR     R0,RP04         ;GO READ HEADER
```

```
                                                FMTDPB                  ;ADRS OF PARAMETER TBL
                                                BR      1$              ;WAIT IF QUEUE IS FULL
                                        2$:     TST     FMTDPB+16       ;WAIT FOR COMMAND TO COMPLETE
                                                BEQ     2$              ;BRANCH IF NOT DONE
                                                BPL     4$              ;BR IF NOT ERROR
                                                MOV     #5$,$ESCAPE     ;;ESCAPE TO 5$ ON ERROR
                                                JSR     PC,ERINDX       ;SEE WHICH ERROR
                                                ERROR   10              ;DRIVE OFFLINE
                                                ERROR   11              ;PERSISTENT DRIVE UNSAFE ERROR
                                                ERROR   12              ;UNCORRECTABLE MASSBUS PARITY ERROR
                                                ERROR   13              ;SOFTWARE TIMEOUT
                                                ERROR   14              ;DRIVE UNSAFE ERROR
                                                BIT     #↑C<HCE>,FMTDPB+16  ;IS ONLY ERROR A HEADER COMPARE ERROR ?
                                                BEQ     3$              ;BR IF YES
                                                ERROR   21              ;CONTROLLER/DRIVE ERROR VERIFYING HEADERS
                                        3$:     CLR     $ESCAPE         ;CLEAR THE ERROR ESCAPE ADDRESS
                                                ERROR   22              ;HEADER COMPARE ERROR VERIFYING HEADERS
                                                JSR     PC,LOP.CK       ;CHECK FOR LOOP ON ERROR
                                        4$:     CMP     RBUF,BUFP       ;SEE IF CYL READ EQUALS CYL EXPECTED
                                                BEQ     6$              ;BR IF CYL CORRECT
                                                ERROR   23              ;CYLINDER NOT CORRECT
                                        5$:     JSR     PC,LOP.CK       ;CHECK FOR LOOP ON ERROR
                                        6$:     CMP     ENDCYL,CYLCK    ;SEE IF LAST CYLINDER
                                                BNE     7$              ;BR IF NOT FINISHED
                                                JMP     $EOP            ;END OF FORMAT
                                        7$:     INC     CYLCK           ;INCREMENT CYLINDER ADDRESS BEING CHECKED
                                                JSR     PC,UPDACY       ;SET UP FOR NEXT CYL
                                                INC     FMTDPB+12       ;ADVANCE TO NEXT CYL #
                                                BR      1$              ;GO READ NEXT HEADER

                                        .SBTTL  END OF PASS ROUTINE

                                        ;;*****************************************************
                                        ;;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;;*IF THERES A MONITOR GO TO IT
                                        ;;*IF THERE ISN'T JUMP TO DONE

                                        $EOP:
                                                TST     MODE            ;'FORMAT' OR 'CHECK' MODE ?
                                                BEQ     3$              ;BR IF 'CHECK'
                                                TYPE    ,MFCMPT         ;'FORMAT COMPLETE'
                                                BR      4$              ;FINISH THE END MESSAGE
                                        3$:     TYPE    ,MCCMPT         ;'CHECK COMPLETE'
                                        4$:     TYPE    ,NUMERR         ;'TOTAL ERRORS DETECTED: '
                                                MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                                TYPE    ,$CRLF          ;CR-LF
                                                INC     $PASS           ;;INCREMENT THE PASS NUMBER
                                                BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
                                                DEC     (PC)+           ;;LOOP?
                                        $EOPCT: .WORD   1
                                                BGT     $DOAGN          ;;YES
                                                MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
                                        $ENDCT: .WORD   1
                                                $EOPCT
                                        $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
```

```
                                        BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
                                        RESET                   ;;CLEAR THE WORLD
                                $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
                                        NOP                     ;;SAVE ROOM
                                        NOP                     ;;FOR
                                        NOP                     ;;ACT11
                                $DOAGN:
                                        JMP     @(PC)+          ;;RETURN
                                $RTNAD: .WORD   DONE
                                DONE:   BIT     #SW0,@SWR       ;SEE IF SWR 0 SET
                                        BNE     1$              ;BR IF SET
                                        JMP     M0              ;ASK FOR NEXT DRIVE
                                1$:     MOV     #STACK,SP       ;RESET STACK POINTER
                                        JMP     M5              ;DO THE FORMAT OR CHECK AGAIN

                                ;;*****************************************************************

                                .SBTTL  SUPPORT SUBROUTINES

                                ;;*****************************************************************

                                ;THIS ROUTINE DETERMINES THE ERROR TYPE

                                ERINDX: MOV     R1,-(SP)        ;STORE R1
                                        CLR     R1              ;CLEAR R1
                                        BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
                                        .WORD   BIT14!BIT13!BIT2!BIT1  ;DRIVE OFFLINE
                                        BNE     5$              ;BR IF OFFLINE
                                        BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
                                        .WORD   BIT12           ;DRIVE PERSISTENTLY UNSAFE
                                        BNE     4$              ;BR IF UNSAFE
                                        BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
                                        .WORD   BIT11!BIT10     ;UNCORRECTABLE MASSBUS PARITY ERROR ?
                                        BNE     3$              ;BR IF PARITY ERROR
                                        BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
                                        .WORD   BIT09!BIT08     ;SOFTWARE TIMEOUT ?
                                        BNE     2$              ;BR IF YES
                                        BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
                                        .WORD   BIT04           ;DRIVE UNSAFE ERROR ?
                                        BNE     1$              ;BR IF YES
                                        INC     R1              ;INCREMENT THE RETURN INDEX
                                1$:     INC     R1              ;INCREMENT THE RETURN INDEX
                                2$:     INC     R1              ;INCREMENT THE RETURN INDEX
                                3$:     INC     R1              ;INCREMENT THE RETURN INDEX
                                4$:     INC     R1              ;INCREMENT THE RETURN INDEX
                                5$:     ASL     R1              ;DOUBLE THE INCREMENT
                                        ADD     R1,2(SP)        ;DEVELOP THE RETURN ADDRESS
                                        MOV     (SP)+,R1        ;RESTORE R1
                                        RTS     PC              ;RETURN

                                ;ROUTINE TO CHECK FOR LOOP ON ERROR

                                LOP.CK: BIT     #SW09,@SWR      ;LOOP ON ERROR ?
                                        BEQ     1$              ;BR IF NOT
                                        JMP     @$LPERR         ;GO TO THE LOOP ON ERROR ADDRESS
                                1$:     CLR     $ESCAPE         ;CLEAR THE ERROR ESCAPE ADDRESS
```

```
1824   005354  033727  001372          BIT     FMTDPB+16,(PC)+   ;CHECK FOR 'FATAL' ERROR
1825   005360  072002                  .WORD   BIT14!BIT13!BIT12!BIT10!BIT01  ;'FATAL' ERROR BITS
1826   005362  001004                  BNE     2$                ;BR IF NOT
1827   005364  032737  004000  001320  BIT     #WLE,RP.REG+RPER  ;WRITE LOCK ERROR ?
1828   005372  001402                  BEQ     3$                ;BR IF NOT
1829   005374  000137  005100  2$:     JMP     $EOP              ;TERMINATE THE FORMAT
1830   005400  000207          3$:     RTS     PC                ;RETURN
1831
1832                                  ;THIS ROUTINE SETS UP THE INPUT TABLE FOR THE DRIVER ROUTINE
1833
1834   005402  113737  001214  001354  SETTBL: MOVB    DRIVE,FMTDPB      ;SET UP DRIVE #
1835   005410  105037  001357          CLRB    FMTDPB+3          ;CLEAR HIGH ORDER ADRS BITS
1836   005414  013737  001260  001360  MOV     MWC,FMTDPB+4      ;LOAD UP WORD COUNT
1837   005422  012737  025130  001362  MOV     #BUFP,FMTDPB+6    ;LOAD UP CURRENT ADRS
1838   005430  105037  001364          CLRB    FMTDPB+10         ;SET SECTOR TO ZERO
1839   005434  113737  001230  001365  MOVB    BEGTRK,FMTDPB+11   ;SET UP STARTING TRK ADRS
1840   005442  013737  001224  001366  MOV     BEGCYL,FMTDPB+12   ;SET UP STARTING CYL
1841   005450  005037  001372          CLR     FMTDPB+16         ;CLEAR RP04 STATUS
1842   005454  013737  001230  001236  MOV     BEGTRK,TRKCNT     ;SET UP PARTIAL CYL TRACK COUNT
1843   005462  013737  001232  001234  MOV     TTRKS,TTRKSC      ;SET UP TOTAL TRACKS COUNTER
1844   005470  000207                  RTS     PC                ;RETURN FROM SETUP
1845
1846                                  ;THIS ROUTINE CONTROLS THE DISK ADDRESSING AND TOTAL TRK COUNT
1847                                  ;IT IS ENTERED AFTER EVERY TRK OPERATION
1848
1849   005472  005337  001234  TRKTST: DEC     TTRKSC            ;SEE IF LAST TRACK
1850   005476  001425                  BEQ     3$                ;BRANCH IF LAST TRACK
1851   005500  022737  000022  001236  CMP     #18.,TRKCNT       ;IS THIS THE LAST TRACK IN THE CYLINDER ?
1852   005506  001407                  BEQ     1$                ;BRANCH IF SO
1853   005510  005237  001236          INC     TRKCNT            ;COUNT UP TRACK WITHIN CYLINDER
1854   005514  105237  001365          INCB    FMTDPB+11         ;INCREMENT TRACK NUMBER
1855   005520  004737  005712          JSR     PC,UPDATK         ;UPDATE TRACK ADDRESS IN BUFFER
1856   005524  000410                  BR      2$                ;EXIT
1857   005526  005037  001236  1$:     CLR     TRKCNT            ;CLEAR TRACK COUNT FOR NEXT CYLINDER
1858   005532  105037  001365          CLRB    FMTDPB+11         ;RESET TRACK ADDRESS TO 0
1859   005536  005237  001366          INC     FMTDPB+12         ;UPDATE CYLINDER NUMBER
1860   005542  004737  005662          JSR     PC,UPDACY         ;UPDATE CYLINDER NUMBER IN BUFFER
1861   005546  062716  000002  2$:     ADD     #2,(SP)           ;ADD TWO TO RETURN ADRS
1862   005552  000207          3$:     RTS     PC                ;RETURN
1863
1864                                  ;THIS ROUTINE SETS UP WC & BA FOR REMAINING SECTORS
1865                                  ;AFTER A WRITE CHECK ERROR
1866
1867   005554  013737  001254  001360  SCAWC:  MOV     SAVWC,FMTDPB+4    ;SET UP WC FOR REMAINING SECTORS
1868   005562  062737  001010  001362          ADD     #520.,FMTDPB+6    ;SET UP BA FOR REMAINING SECTORS
1869   005570  005237  001364          INC     FMTDPB+10         ;ADVANCE TO NEXT SECTOR IN TBL
1870   005574  005037  001216          CLR     SOFSW             ;RESET RETRY COUNTER
1871   005600  000207                  RTS     PC                ;RETURN TO COMPLETE TRK FORMAT
1872
1873
1874                                  ;THIS ROUTINE SETS UP THE CYLINDER ADRS, FORMAT BIT, TRACK AND
1875                                  ;SECTOR ADRS IN MEMORY WITH THE STARTING CYL - TRK INFORMATION
1876
1877   005602  013701  001266  SETHDR: MOV     MAXSEC,R1         ;SET UP SECTOR COUNT
1878   005606  012700  025130          MOV     #BUFP,R0          ;SET UP HEADER POINTER IN R0
1879   005612  013702  001224          MOV     BEGCYL,R2         ;PUT STARTING CYL # IN R2
```

```
1880  005616  013703  001230          MOV     BEGTRK,R3       ;PUT STARTING TRK # IN R3
1881  005622  000303                  SWAB    R3              ;JUSTIFY TRACK ADRS
1882  005624  005737  001264          TST     SEC20           ;SEE IF 20 OR 22 SECTOR MODE
1883  005630  001402                  BEQ     1$              ;BR IF 20 SECTOR MODE
1884  005632  052702  010000          BIS     #10000,R2       ;SET THE 22 SECTOR FORMAT BIT
1885  005636  010220          1$:     MOV     R2,(R0)+        ;WRITE IN HEADER AREA OF CORE THE CYL ADRS
1886  005640  010320                  MOV     R3,(R0)+        ;WRITE IN HEADER AREA OF CORE THE TRK ADRS
1887  005642  005020                  CLR     (R0)+           ;CLR 1ST KEYWORD
1888  005644  005010                  CLR     (R0)            ;CLR 2ND KEYWORD
1889  005646  062700  001002          ADD     #514.,R0        ;SET UP FOR NEXT HEADER
1890  005652  005203                  INC     R3              ;UPDATE SECTOR ADRS FOR NEXT HEADER
1891  005654  005301                  DEC     R1              ;MAINTAIN COUNT OF SECTORS
1892  005656  002367                  BGE     1$              ;BRANCH IF NOT LAST SECTOR
1893  005660  000207                  RTS     PC              ;EXIT - HEADERS ARE LOADED INTO CORE
1894
1895                          ;THIS ROUTINE UPDATES THE CYLINDER ADRS OF THE HEADER WORDS IN CORE
1896
1897  005662  013701  001266  UPDACY: MOV     MAXSEC,R1       ;SET UP SECTOR COUNT
1898  005666  012700  025130          MOV     #BUFP,R0        ;SET UP HEADER POINTER IN R0
1899  005672  005220          1$:     INC     (R0)+           ;INCREMENT FOR NEXT CYLINDER
1900  005674  042710  177400          BIC     #177400,(R0)    ;RESET TRK ADRS TO 0
1901  005700  062700  001006          ADD     #518.,R0        ;SET UP FOR NEXT HEADER
1902  005704  005301                  DEC     R1              ;COUNT SECTORS
1903  005706  002371                  BGE     1$              ;BRANCH IF NOT LAST SECTOR
1904  005710  000207                  RTS     PC              ;EXIT
1905
1906                          ;THIS ROUTINE UPDATES THE TRACK ADRS OF THE HEADER WORDS IN CORE
1907
1908  005712  013701  001266  UPDATK: MOV     MAXSEC,R1       ;SET UP SECTOR COUNT
1909  005716  012700  025130          MOV     #BUFP,R0        ;SET UP HEADER POINTER IN R0
1910  005722  005720                  TST     (R0)+           ;POINT HEADER POINTER TO TRK - SEC ADRS
1911  005724  062710  000400  1$:     ADD     #400,(R0)       ;INDEX TRK ADRS
1912  005730  062700  001010          ADD     #520.,R0        ;SET UP FOR NEXT HEADER
1913  005734  005301                  DEC     R1              ;COUNT SECTORS
1914  005736  002372                  BGE     1$              ;BRANCH IF NOT LAST SECTOR
1915  005740  000207                  RTS     PC              ;EXIT
1916
1917                          ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
1918
1919  005742  012737  006020  000004  ST.CLK: MOV     #STCLK1,@#ERRVEC ;SET UP VECTOR FOR P CLK
1920  005750  005037  000006          CLR     @#ERRVEC+2      ;NEW PSW
1921  005754  005777  173216          TST     @$LKCSR         ;CHECK FOR KW11-P
1922  005760  013746  001202          MOV     $LPVEC,-(SP)    ;VECTOR ADDRESS
1923  005764  012776  006104  000000  MOV     #CLOCK,@(SP)    ;SET UP KW11-P VECTOR
1924  005772  062716  000002          ADD     #2,(SP)         ;POINT TO PSW
1925  005776  012736  000300          MOV     #PR6,@(SP)+     ;PSW - PRI 6
1926  006002  012777  177777  173170  MOV     #-1,@$LKCSB     ;LOAD COUNTER BUFFER
1927  006010  012777  000135  173160  MOV     #135,@$LKCSR    ;SET CLK - CNT UP
1928  006016  000426                  BR      STCLK3
1929  006020  062706  000004  STCLK1: ADD     #4,SP           ;RESTORE THE STACK POINTER
1930  006024  012737  006070  000004  MOV     #STCLK2,@#ERRVEC        ;CHANGE ERROR VECTOR
1931  006032  005777  173150          TST     @$LKS           ;LOOK FOR KW11-L
1932  006036  013746  001210          MOV     $LLVEC,-(SP)    ;KW11-L VECTOR ADDRESS
1933  006042  012776  006104  000000  MOV     #CLOCK,@(SP)    ;SET UP KW11-L VECTOR
1934  006050  062716  000002          ADD     #2,(SP)         ;INCREMENT VECTOR ADDRESS
1935  006054  012736  000300          MOV     #PR6,@(SP)+     ;PSW - PRI 6
```

```
1936  006060  012777  000100  173120          MOV     #100,@$LKS      ;SET KW11-L INTERRUPT ENABLE
1937  006066  000402                          BR      STCLK3
1938  006070  062706  000004          STCLK2: ADD     #4,SP           ;RESTORE THE STACK POINTER
1939  006074  012737  000006  000004  STCLK3: MOV     #6,@#ERRVEC     ;RESTORE THE ERROR VECTOR
1940  006102  000207                          RTS     PC
1941
1942                                  ;THIS CODE SERVICES A CLOCK INTERRUPT EVERY 16MS
1943
1944  006104  012746  000020          CLOCK:  MOV     #16.,-(SP)      ;PUT MILLISECONDS ON THE STACK
1945  006110  004737  016466                  JSR     PC,RPTMR        ;GO REPORT TIME
1946  006114  000002                          RTI                     ;RETURN AND CONTINUE
1947
1948                                  ;ROUTINE TO INTERCEPT 'CONTROL C' TYPED DURING PARAMETER ENTRY TIME
1949
1950  006116  012706  001100          OENTER: MOV     #STACK,SP       ;INITIALIZE THE STACK
1951  006122  005737  012160          1$:     TST     TRNSWT          ;ALL ACTIVITY STOPPED ?
1952  006126  001375                          BNE     1$              ;BR IF NOT
1953  006130  000005                          RESET                   ;CLEAR THE BUS
1954  006132  000137  003062                  JMP     M1B             ;START AGAIN WITH DRIVE SELECTION
1955
1956                                  ;ROUTINE TO TYPE THE PRESENT DISK ADDRESS.  ENTERED BY TYPING 'CONTROL C'
1957
1958  006136  005037  177776          TYPADR: CLR     PSW             ;SET PROCESSOR TO PRIORITY 0
1959  006142  012737  006116  001300          MOV     #OENTER,CNTLC   ;CHANGE 'CONTROL C' RETURN ADDRESS
1960  006150  104401  022065                  TYPE    ,ADDRIS         ;'PRESENT ADDRESS IS: '
1961  006154  104401  020675                  TYPE    ,C              ;'C'
1962  006160  013746  001366                  MOV     FMTDPB+12,-(SP) ;PUT THE CYLINDER ADDRESS ON THE STACK
1963  006164  004737  011470                  JSR     PC,$SB2D        ;CONVERT IT TO DECIMAL
1964  006170  004737  011430                  JSR     PC,$SUPRS       ;TYPE IT
1965  006174  104401  020705                  TYPE    ,LINSP          ;SPACES
1966  006200  104401  020677                  TYPE    ,T              ;'T'
1967  006204  005046                          CLR     -(SP)           ;CLEAR THE STACK
1968  006206  113716  001365                  MOVB    FMTDPB+11,(SP)  ;PUT THE TRACK ADDRESS ON THE STACK
1969  006212  004737  011470                  JSR     PC,$SB2D        ;CONVERT IT TO DECIMAL
1970  006216  004737  011430                  JSR     PC,$SUPRS       ;TYPE IT
1971  006222  104401  001167                  TYPE    ,$CRLF          ;CR-LF
1972  006226  012737  006136  001300          MOV     #TYPADR,CNTLC   ;RESTORE ENTRANCE TO THIS ROUTINE
1973  006234  000002                          RTI                     ;RETURN
1974
1975                                  ;PARAMETER ENTRY ROUTINE
1976                                  ;CALL
1977                                  ;       MOV     #ADR,R3         ;PARAMETER TABLE ADDRESS
1978                                  ;       JSR     PC,PARENT       ;GET THE PARAMETERS
1979
1980  006236  010346          PARENT: MOV     R3,-(SP)        ;SAVE R3
1981  006240  012703  001374          MOV     #TABLE,R3       ;PARAMETER TABLE ADDRESS
1982  006244  012337  006254  1$:     MOV     (R3)+,3$        ;ADDRESS OF PARAMETER NAME
1983  006250  001436                  BEQ     9$              ;BR IF AT END OF TABLE
1984  006252  104401                  TYPE                    ;TYPE THE PARAMETER NAME
1985  006254  000000          3$:     .WORD   0               ;ADDRESS OF PARAMETER NAME TEXT
1986  006256  012302                  MOV     (R3)+,R2        ;MAXIMUM PARAMETER VALUE
1987  006260  012305                  MOV     (R3)+,R5        ;ADDRESS OF PARAMETER
1988  006262  011546                  MOV     (R5),-(SP)      ;CURRENT VALUE OF PARAMETER
1989  006264  104405                  TYPOS                   ;TYPE THE CURRENT VALUE OF THE PARAMETER
1990  006266  104401  020671          TYPE    ,SLASH          ;'/'
1991  006272  104411                  RDLIN                   ;READ THE KEYBOARD
```

SUPPORT SUBROUTINES


```
1992  006274  012601                  MOV     (SP)+,R1        ;INPUT ASCII STRING ADDRESS
1993  006276  004537  006500          JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
1994  006302  006244                  1$                      ;CARRIAGE RETURN ONLY ENTERED
1995  006304  006346                  9$                      ;PERIOD ONLY ENTERED
1996  006306  006322                  6$                      ;ILLEGAL INPUT
1997  006310  006316                  5$                      ;TERMINATED WITH A CARRIAGE RETURN
1998  006312  006322                  6$                      ;TERMINATED WITH A "."
1999  006314  006334                  7$                      ;TERMINATED WITH A ","
2000  006316  010215          5$:     MOV     R2,(R5)         ;MOVE NEW VALUE TO PARAMETER LOCATION
2001  006320  000751                  BR      1$              ;GET MORE PARAMETERS
2002  006322  104401  021436  6$:     TYPE    ,BADENT         ;'BAD ENTRY'
2003  006326  162703  000006          SUB     #6,R3           ;DECREMENT THE TABLE POINTER
2004  006332  000744                  BR      1$              ;TRY AGAIN
2005  006334  010215          7$:     MOV     R2,(R5)         ;NEW VALUE
2006  006336  000403                  BR      9$              ;EXIT
2007  006340  012703  001374  8$:     MOV     #TABLE,R3       ;RELOAD THE PARAMETER TABLE ADDRESS
2008  006344  000737                  BR      1$              ;TRY AGAIN
2009  006346  012603          9$:     MOV     (SP)+,R3        ;RESTORE R3
2010  006350  000207                  RTS     PC              ;RETURN
2011
2012
2013                                  ;THIS ROUTINE IS USED TO CHECK IF AN
2014                                  ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
2015                                  ;CALL
2016                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
2017                                  ;       JSR     R5,CK.OCT       ;CHECK THE CHARACTER
2018                                  ;       RETURN1                 ;CHARACTER IS NOT BETWEEN 0-7
2019                                  ;       RETURN2                 ;CHARACTER IS IN R2 AS A
2020                                  ;                               ;OCTAL DIGIT
2021
2022  006352  121127  000060  CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
2023  006356  103407                  BLO     1$              ;YES -- BRANCH
2024  006360  121127  000067          CMPB    (R1),#'7        ;GREATER THAN SEVEN?
2025  006364  101004                  BHI     1$              ;YES -- BRANCH
2026  006366  111102                  MOVB    (R1),R2         ;GET THE CHARACTER
2027  006370  042702  177770          BIC     #^C7,R2         ;STRIP AWAY THE ASCII
2028  006374  005725                  TST     (R5)+           ;ADJUST FOR RETURN
2029  006376  000205          1$:     RTS     R5              ;RETURN
2030
2031                                  ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
2032                                  ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
2033                                  ;CALL
2034                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
2035                                  ;       JSR     R5,CK.DEC       ;CHECK THE CHARACTER
2036                                  ;       RETURN1                 ;NOT BETWEEN 0 AND 9
2037                                  ;       RETURN2                 ;BETWEEN 0 AND 9
2038                                  ;                               ;R2 = DIGIT
2039
2040  006400  121127  000060  CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
2041  006404  103407                  BLO     1$              ;YES -- BRANCH
2042  006406  121127  000071          CMPB    (R1),#'9        ;GREATER THAN NINE?
2043  006412  101004                  BHI     1$              ;YES -- BRANCH
2044  006414  111102                  MOVB    (R1),R2         ;GET THE CHARACTER
2045  006416  042702  000060          BIC     #'0,R2          ;STRIP AWAY THE ASCII
2046  006422  005725                  TST     (R5)+           ;ADJUST FOR RETURN
2047  006424  000205          1$:     RTS     R5              ;RETURN
```

```
2048
2049                                    ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
2050                                    ;DETERMINE WHAT IT IS.
2051                                    ;CALL
2052                                    ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
2053                                    ;       JSR     R5,CK.CHR       ;CHECK CHARACTER
2054                                    ;       RETURN  ADR1            ;UNKNOWN CHARACTER
2055                                    ;       RETURN  ADR2            ;CARRIAGE RETURN * (R1)=ADR+1
2056                                    ;       RETURN  ADR3            ;COMMA * (R1)=ADR+1
2057                                    ;       RETURN  ADR4            ;PERIOD * (R1)=ADR+1
2058                                    ;       RETURN  ADR5            ;DIGIT BETWEEN 0 AND 7.
2059                                    ;       RETURN  ADR6            ;DIGIT BETWEEN 8 AND 9.
2060                                    ;                               ;R2 = DIGIT * (R1)=ADR+1
2061
2062  006426  105711            CK.CHR: TSTB    (R1)                    ;"CARRIAGE RETURN"?
2063  006430  001417                    BEQ     3$                      ;YES -- BRANCH
2064  006432  121127  000054            CMPB    (R1),#',                ;"COMMA"?
2065  006436  001413                    BEQ     2$                      ;YES -- BRANCH
2066  006440  121127  000056            CMPB    (R1),#'.                ;"PERIOD"?
2067  006444  001407                    BEQ     1$                      ;YES -- BRANCH
2068  006446  004537  006400            JSR     R5,CK.DEC               ;"DIGIT"?
2069  006452  000410                    BR      4$                      ;NO -- BRANCH
2070  006454  004537  006352            JSR     R5,CK.OCT               ;OCTAL ?
2071  006460  005725                    TST     (R5)+                   ;DIGIT BETWEEN 8-9
2072  006462  005725                    TST     (R5)+                   ;DIGIT BETWEEN 0-7
2073  006464  005725            1$:     TST     (R5)+                   ;PERIOD
2074  006466  005725            2$:     TST     (R5)+                   ;COMMA
2075  006470  005725            3$:     TST     (R5)+                   ;CARRIAGE RETURN
2076  006472  005201                    INC     R1                      ;MOVE POINTER TO NEXT CHARACTER
2077  006474  011505            4$:     MOV     (R5),R5                 ;UNKNOWN CHARACTER
2078  006476  000205                    RTS     R5                      ;RETURN
2079
2080                                    ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
2081                                    ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
2082                                    ;CALL
2083                                    ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
2084                                    ;       MOV     #NUM,R2         ;MAX. MAGNITUDE OF INPUT NUMBER
2085                                    ;       JSR     R5,CK.DIG       ;CHECK DIGITS
2086                                    ;       RETURN  ADR1            ;"CR" ONLY ENTERED -- R2=0
2087                                    ;       RETURN  ADR2            ;"PERIOD" ONLY ENTERED -- R2=0
2088                                    ;       RETURN  ADR3            ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
2089                                    ;       RETURN  ADR4            ;"CR" -- R2 = NUMBER
2090                                    ;       RETURN  ADR5            ;"COMMA" -- R2 = NUMBER
2091                                    ;       RETURN  ADR6            ;"PERIOD" -- R2 = NUMBER
2092
2093  006500  010446            CK.DIG: MOV     R4,-(SP)                ;SAVE R4
2094  006502  010346                    MOV     R3,-(SP)                ;SAVE R3
2095  006504  010246                    MOV     R2,-(SP)                ;SAVE THE MAX. SIZE ON THE STACK
2096  006506  005002                    CLR     R2                      ;START WITH 0
2097  006510  005003                    CLR     R3
2098  006512  005004                    CLR     R4
2099  006514  004537  006426            JSR     R5,CK.CHR               ;CHECK ONE CHARACTER
2100  006520  006614                    6$                              ;ILLEGAL CHARACTER
2101  006522  006622                    9$                              ;CARRIAGE RETURN
2102  006524  006614                    6$                              ;" "
2103  006526  006616                    7$                              ;"."
```

```
2104  006530  006534                          1$                      ;DIGIT 0-7
2105  006532  006534                          1$                      ;DIGIT 8-9
2106  006534  062705  000004          1$:     ADD     #4,R5           ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
2107  006540  006303                  2$:     ASL     R3              ;INPUT NUMBER *2
2108  006542  010346                          MOV     R3,-(SP)        ;SAVE *2
2109  006544  006303                          ASL     R3              ;*4
2110  006546  006303                          ASL     R3              ;*8
2111  006550  062603                          ADD     (SP)+,R3        ;(*2)+(*8) = *10
2112  006552  060203                          ADD     R2,R3           ;UPDATE THE INPUT NUMBER
2113  006554  004537  006426                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
2114  006560  006620                          8$                      ;ILLEGAL CHARACTER
2115  006562  006604                          5$                      ;CARRIAGE RETURN
2116  006564  006602                          4$                      ;","
2117  006566  006574                          3$                      ;"."
2118  006570  006540                          2$                      ;DIGIT 0-7
2119  006572  006540                          2$                      ;DIGIT 8-9
2120  006574  105711                  3$:     TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
2121  006576  001010                          BNE     8$              ;BR IF NOT
2122  006600  005724                          TST     (R4)+           ;INCREMENT THE RETURN
2123  006602  005724                  4$:     TST     (R4)+           ;INCREMENT THE RETURN
2124  006604  005724                  5$:     TST     (R4)+           ;INCREMENT THE RETURN
2125  006606  020316                          CMP     R3,(SP)         ;CHECK THE MAGNITUDE OF THE NUMBER
2126  006610  101004                          BHI     9$              ;BR IF ENTERED NUMBER TOO LARGE
2127  006612  000402                          BR      8$              ;BYPASS INCREMENT
2128  006614  005725                  6$:     TST     (R5)+           ;INCREMENT RETURN PAST INVALID RETURN
2129  006616  005725                  7$:     TST     (R5)+           ;INCREMENT RETURN
2130  006620  060405                  8$:     ADD     R4,R5           ;SETUP RETURN POINTER
2131  006622  010302                  9$:     MOV     R3,R2           ;ENTERED VALUE
2132  006624  005726                          TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
2133  006626  012603                          MOV     (SP)+,R3        ;RESTORE R3
2134  006630  012604                          MOV     (SP)+,R4        ;RESTORE R4
2135  006632  011505                          MOV     (R5),R5         ;GET RETURN ADDRESS
2136  006634  000205                          RTS     R5              ;RETURN
2137
2138                                  .SBTTL  MACRO ROUTINES
2139
2140
2141                                  .SBTTL  ERROR HANDLER ROUTINE
2142
2143                                  ;;*****************************************************************
2144                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2145                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2146                                  ;*AND GO TO TYPERR ON ERROR
2147                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2148                                  ;*SW15=1         HALT ON ERROR
2149                                  ;*SW13=1         INHIBIT ERROR TYPEOUTS
2150                                  ;*SW10=1         BELL ON ERROR
2151                                  ;*SW09=1         LOOP ON ERROR
2152                                  ;*CALL
2153                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2154
2155  006636                          $ERROR:
2156  006636  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
2157  006640  010137  001274                  MOV     R1,DDRIVE       ;DRIVE ADDRESS IF DRIVE ERROR CALL
2158  006644  010337  001276                  MOV     R3,ATTN         ;ATTENTION REGISTER CONTENTS
2159  006650  013737  001366  001270          MOV     FMTDPB+12,DS.CYL  ;CURRENT CYLINDER ADDRESS
```

```
2160  006656  113737  001365  001272          MOVB    FMTDPB+11,DS.TRK  ;REQUESTED TRACK ADDRESS
2161  006664  005737  001310                  TST     RP.REG+RPBA       ;NON-ZERO BUFFER ADDRESS ?
2162  006670  001406                          BEQ     7$                ;BR IF NO BUFFER ADDRESS
2163  006672  013746  001310                  MOV     RP.REG+RPBA,-(SP) ;BUFFER ADDRESS
2164  006676  162716  000002                  SUB     #2,(SP)           ;DECREMENT THE ADDRESS
2165  006702  013637  001124                  MOV     @(SP)+,$GDDAT     ;GET THE BUFFER WORD WHICH DIDN'T COMPARE
2166  006706  105237  001103          7$:     INCB    $ERFLG            ;;SET THE ERROR FLAG
2167  006712  001775                          BEQ     7$                ;;DON'T LET THE FLAG GO TO ZERO
2168  006714  013777  001102  172220          MOV     $TSTNM,@DISPLAY   ;;DISPLAY TEST NUMBER AND ERROR FLAG
2169  006722  032777  002000  172210          BIT     #BIT10,@SWR       ;;BELL ON ERROR?
2170  006730  001402                          BEQ     1$                ;;NO - SKIP
2171  006732  104401  001162                  TYPE    ,$BELL            ;;RING BELL
2172  006736  005237  001112          1$:     INC     $ERTTL            ;;COUNT THE NUMBER OF ERRORS
2173  006742  011637  001116                  MOV     (SP),$ERRPC       ;;GET ADDRESS OF ERROR INSTRUCTION
2174  006746  162737  000002  001116          SUB     #2,$ERRPC
2175  006754  117737  172136  001114          MOVB    @$ERRPC,$ITEMB    ;;STRIP AND SAVE THE ERROR ITEM CODE
2176  006762  032777  020000  172150          BIT     #BIT13,@SWR       ;;SKIP TYPEOUT IF SET
2177  006770  001004                          BNE     20$               ;;SKIP TYPEOUTS
2178  006772  004737  007044                  JSR     PC,TYPERR         ;;GO TO USER ERROR ROUTINE
2179  006776  104401  001167                  TYPE    ,$CRLF
2180  007002                          20$:
2181  007002  005777  172132          2$:     TST     @SWR              ;;HALT ON ERROR
2182  007006  100002                          BPL     3$                ;;SKIP IF CONTINUE
2183  007010  000000                          HALT                      ;;HALT ON ERROR!
2184  007012  104407                          CKSWR                     ;;TEST FOR CHANGE IN SOFT-SWR
2185  007014  032777  001000  172116  3$:     BIT     #BIT09,@SWR       ;;LOOP ON ERROR SWITCH SET?
2186  007022  001402                          BEQ     4$                ;;BR IF NO
2187  007024  013716  001110                  MOV     $LPERR,(SP)       ;;FUDGE RETURN FOR LOOPING
2188  007030  005737  001160          4$:     TST     $ESCAPE           ;;CHECK FOR AN ESCAPE ADDRESS
2189  007034  001402                          BEQ     5$                ;;BR IF NONE
2190  007036  013716  001160                  MOV     $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
2191  007042                          5$:
2192  007042  000002                          RTI                       ;;RETURN
2193
2194
2195                                  ;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
2196                                  ;WHICH ERROR IS TO BE REPORTED, IT THEN OBTAINS, FROM THE "ERROR
2197                                  ;TABLE" ($ERRTB), AND REPORTS THE APPROPIATE INFORMATION
2198                                  ;CONCERNING THE ERROR.
2199
2200  007044  104412                  TYPERR: SAVREG                    ;SAVE R0-R5
2201  007046  005000                          CLR     R0                ;CLEAR R0 FOR ERROR NUMBER
2202  007050  113700  001114                  MOVB    $ITEMB,R0         ;ERROR NUMBER
2203  007054  005300                          DEC     R0                ;FORM INDEX FOR ERROR TABLE
2204  007056  006300                          ASL     R0
2205  007060  006300                          ASL     R0
2206  007062  006300                          ASL     R0
2207  007064  062700  001626          1$:     ADD     #$ERRTB,R0        ;FORM ADDRESS
2208  007070  012037  007104                  MOV     (R0)+,2$          ;GET ERROR MESSAGE (EM) POINTER
2209  007074  001404                          BEQ     3$                ;BRANCH IF THERE ISN'T ONE
2210  007076  104401  001167                  TYPE    ,$CRLF            ;"CARRIAGE RETURN - LINE FEED
2211  007102  104401                          TYPE
2212  007104  000000                  2$:     .WORD   0                 ;"EM" POINTER GOES HERE
2213  007106  012037  007122          3$:     MOV     (R0)+,4$          ;PICK UP DATA HEADER (DH) POINTER
2214  007112  001404                          BEQ     5$                ;BRANCH IF NONE
2215  007114  104401  001167                  TYPE    ,$CRLF            ;CARRIAGE RETURN-LINE FEED
```

```
                                        TYPE
2216  007120  104401                    4$:     .WORD   0               ;"DH" POINTER GOES HERE
2217  007122  000000                    5$:     MOV     (R0)+,R1        ;PICKUP DATA TABLE (DT) POINTER
2218  007124  012001                            BEQ     20$             ;BRANCH IF NONE
2219  007126  001460                            CLR     R5              ;SET INDENT SWITCH
2220  007130  005005                            MOV     (R0)+,R0        ;DATA FORMAT (DF) POINTER
2221  007132  012000                            MOV     (R0)+,R2        ;NUMBER OF DH'S TO TYPE
2222  007134  012002                            BEQ     17$             ;BRANCH IF DH NUMBER IS 0
2223  007136  001451                            COM     R5              ;NO INDENT
2224  007140  005105                            TYPE    $CRLF           ;CARRIAGE RETURN-LINE FEED
2225  007142  104401  001167                10$:    MOVB    (R0)+,R3        ;NUMBER OF DATA WORDS TO TYPE
2226  007146  112003                            MOVB    (R0)+,R4        ;AND HOW TO TYPE THEM
2227  007150  112004                    11$:    ROR     R4              ;OCTAL OR DECIMAL?
2228  007152  006004                            BCS     12$             ;DECIMAL--BRANCH
2229  007154  103403                            MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
2230  007156  013146                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2231  007160  104402                            BR      13$
2232  007162  000402                    12$:
2233  007164                                    MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
2234  007164  013146                            TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
2235  007166  104405                    13$:    DEC     R3              ;MORE NUMBERS TO TYPE?
2236  007170  005303                            BEQ     14$             ;NO--BRANCH
2237  007172  001403                            TYPE    $LINSP          ;YES--TYPE SEPERATORS
2238  007174  104401  020705                    BR      11$             ;LOOP
2239  007200  000764                    14$:    DEC     R2              ;MORE DH'S?
2240  007202  005302                            BLE     20$             ;NO--BRANCH
2241  007204  003431                            TYPE    $CRLF           ;YES--START A NEW LINE
2242  007206  104401  001167                    TST     2(R0)           ;ONLY A 'DH' IN THIS REQUEST ?
2243  007212  005760  000002                    BEQ     15$             ;BR IF YES - BYPASS THE INDENT
2244  007216  001404                            COM     R5              ;INDENT?
2245  007220  005105                            BNE     15$             ;NO--BRANCH
2246  007222  001002                            TYPE    $LINSP          ;YES--TYPE SPACES
2247  007224  104401  020705            15$:    MOV     (R0)+,16$       ;GET NEXT DH
2248  007230  012037  007236                    TYPE                    ;AND TYPE IT
2249  007234  104401                    16$:    .WORD   0               ;DH POINTER GOES HERE
2250  007236  000000                            TST     (R0)            ;TYPE A 'DT' ?
2251  007240  005710                            BNE     21$             ;BR IF A 'DT'
2252  007242  001003                            ADD     #4,R0           ;INCREMENT THE 'DF' POINTER
2253  007244  062700  000004                    BR      14$             ;SEE IF END OF 'DF' BLOCK
2254  007250  000754                    21$:    TYPE    $CRLF           ;CARRIAGE RETURN-LINE FEED
2255  007252  104401  001167                    TST     R5              ;INDENT?
2256  007256  005705                            BNE     10$             ;NO--BRANCH
2257  007260  001332                    17$:    TYPE    $LINSP          ;YES--TYPE SPACES
2258  007262  104401  020705                    BR      10$             ;LOOP
2259  007266  000727                    20$:    RESREG                  ;RESTORE R0-R5
2260  007270  104413                            RTS     PC              ;RETURN
2261  007272  000207
2262
2263                                    .SBTTL  TYPE ROUTINE
2264
2265                                    ;;******************************************************************
2266                                    ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2267                                    ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2268                                    ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2269                                    ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2270                                    ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2271                                    ;*
```

```
2272                                    ;*CALL:
2273                                    ;*1) USING A TRAP INSTRUCTION
2274                                    ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2275                                    ;*OR
2276                                    ;*      TYPE
2277                                    ;*      MESADR
2278                                    ;*
2279
2280  007274  105737  001157      $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
2281  007300  100002                      BPL     1$              ;;BR IF YES
2282  007302  000000                      HALT                    ;;HALT HERE IF NO TERMINAL
2283  007304  000407                      BR      3$              ;;LEAVE
2284  007306  010046              1$:     MOV     R0,-(SP)        ;;SAVE R0
2285  007310  017600  000002              MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
2286  007314  112046              2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
2287  007316  001005                      BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
2288  007320  005726                      TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
2289  007322  012600              60$:    MOV     (SP)+,R0        ;;RESTORE R0
2290  007324  062716  000002      3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
2291  007330  000002                      RTI                     ;;RETURN
2292  007332  122716  000011      4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
2293  007336  001430                      BEQ     8$
2294  007340  122716  000200              CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
2295  007344  001006                      BNE     5$
2296  007346  005726                      TST     (SP)+           ;;POP  <CR><LF> EQUIV
2297  007350  104401                      TYPE                    ;;TYPE A CR AND LF
2298  007352  001167                      $CRLF
2299  007354  105037  007510              CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
2300  007360  000755                      BR      2$              ;;GET NEXT CHARACTER
2301  007362  004737  007444      5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
2302  007366  123726  001156      6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
2303  007372  001350                      BNE     2$              ;;IF NO GO GET NEXT CHAR.
2304  007374  013746  001154              MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
2305                                                              ;;AND THE NULL CHAR.
2306  007400  105366  000001      7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
2307  007404  002770                      BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
2308  007406  004737  007444              JSR     PC,$TYPEC       ;;GO TYPE A NULL
2309  007412  105337  007510              DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
2310  007416  000770                      BR      7$              ;;LOOP
2311
2312                                    ;HORIZONTAL TAB PROCESSOR
2313
2314  007420  112716  000040      8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
2315  007424  004737  007444      9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
2316  007430  132737  000007  007510      BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
2317  007436  001372                      BNE     9$              ;;TAB STOP
2318  007440  005726                      TST     (SP)+           ;;POP SPACE OFF STACK
2319  007442  000724                      BR      2$              ;;GET NEXT CHARACTER
2320  007444  105777  171500      $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
2321  007450  100375                      BPL     $TYPEC
2322  007452  116677  000002  171472      MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2323  007460  122766  000015  000002      CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
2324  007466  001003                      BNE     1$              ;;BRANCH IF NO
2325  007470  105037  007510              CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
2326  007474  000406                      BR      $TYPEX          ;;EXIT
2327  007476  122766  000012  000002  1$: CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
```

```
                                        BEQ     $TYPEX          ;;BRANCH IF YES
                                        INCB    (PC)+           ;;COUNT THE CHARACTER
                                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
                                $TYPEX: RTS     PC


                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                ;*      .BYTE   M               ;;M=1 OR 0
                                ;*                                      ;;1=TYPE LEADING ZEROS
                                ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                ;*
                                ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                ;*$TYPOS OR $TYPOC
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPON                   ;;CALL FOR TYPEOUT
                                ;*
                                ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOC                   ;;CALL FOR TYPEOUT

                                $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
                                        MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
                                        MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
                                        ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
                                        BR      $TYPON
                                $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
                                        MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
                                $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
                                        MOV     R3,-(SP)        ;;SAVE R3
                                        MOV     R4,-(SP)        ;;SAVE R4
                                        MOV     R5,-(SP)        ;;SAVE R5
                                        MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
                                        NEG     R4
                                        ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
                                        MOVB    R4,$OMODE       ;;SAVE IT FOR USE
                                        MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
                                        MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
                                        CLR     R3              ;;CLEAR THE OUTPUT WORD
                                1$:     ROL     R5              ;;ROTATE MSB INTO "C"
                                        BR      3$              ;;GO DO MSB
                                2$:     ROL     R5              ;;FORM THIS DIGIT
                                        ROL     R5
                                        ROL     R5
                                        MOV     R5,R3
                                3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
```

```
                                        DECB    $OMODE          ;;TYPE THIS DIGIT?
                                        BPL     7$              ;;BR IF NO
                                        BIC     #177770,R3      ;;GET RID OF JUNK
                                        BNE     4$              ;;TEST FOR 0
                                        TST     R4              ;;SUPPRESS THIS 0?
                                        BEQ     5$              ;;BR IF YES
                                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
                                        BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
                                5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
                                        MOVB    R3,8$           ;;SAVE FOR TYPING
                                        TYPE    ,8$             ;;GO TYPE THIS DIGIT
                                7$:     DECB    $OCNT           ;;COUNT BY 1
                                        BGT     2$              ;;BR IF MORE TO DO
                                        BLT     6$              ;;BR IF DONE
                                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
                                        BR      2$              ;;GO DO THE LAST DIGIT
                                6$:     MOV     (SP)+,R5        ;;RESTORE R5
                                        MOV     (SP)+,R4        ;;RESTORE R4
                                        MOV     (SP)+,R3        ;;RESTORE R3
                                        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
                                        MOV     (SP)+,(SP)
                                        RTI                     ;;RETURN
                                8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
                                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
                                $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
                                $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
                                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE

                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

                                ;;**************************************************************
                                ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                ;;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                ;;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                ;;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                ;;*REPLACED WITH SPACES.
                                ;;*CALL:
                                ;;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                ;;*      TYPDS                   ;;GO TO THE ROUTINE

                                $TYPDS:
                                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
                                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
                                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
                                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
                                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
                                        MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
                                        MOV     20(SP),R5       ;;GET THE INPUT NUMBER
                                        BPL     1$              ;;BR IF INPUT IS POS.
                                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
                                        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
                                1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
                                        MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
                                        MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
                                2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
                                        MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
```

```
002440  010016  160105                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
002441  010020  002402                        BLT     4$              ;;BR IF DONE
002442  010022  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
002443  010024  000774                        BR      3$
002444  010026  060105                4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
002445  010030  005702                        TST     R2              ;;CHECK IF BCD DIGIT=0
002446  010032  001002                        BNE     5$              ;;FALL THROUGH IF 0
002447  010034  105716                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
002448  010036  100407                        BMI     7$              ;;BR IF YES
002449  010040  106316                5$:     ASLB    (SP)            ;;MSD?
002450  010042  103003                        BCC     6$              ;;BR IF NO
002451  010044  116663  000001  177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
002452  010052  052702  000060        6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
002453  010056  052702  000040        7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
002454  010062  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
002455  010064  005720                        TST     (R0)+           ;;JUST INCREMENTING
002456  010066  020027  000010                CMP     R0,#10          ;;CHECK THE TABLE INDEX
002457  010072  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
002458  010074  003002                        BGT     8$              ;;GO TO EXIT
002459  010076  010502                        MOV     R5,R2           ;;GET THE LSD
002460  010100  000764                        BR      6$              ;;GO CHANGE TO ASCII
002461  010102  105726                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
002462  010104  100003                        BPL     9$              ;;BR IF NO
002463  010106  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
002464  010114  105013                9$:     CLRB    (R3)            ;;SET THE TERMINATOR
002465  010116  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
002466  010120  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
002467  010122  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
002468  010124  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
002469  010126  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
002470  010130  104401  010156                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
002471  010134  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
002472  010142  012616                        MOV     (SP)+,(SP)
002473  010144  000002                        RTI                     ;;RETURN TO USER
002474  010146  023420                $DTBL:  10000.
002475  010150  001750                        1000.
002476  010152  000144                        100.
002477  010154  000012                        10.
002478  010156  000004                $DBLK:  .BLKW   4
002479
002480                                  .SBTTL  TTY INPUT ROUTINE
002481
002482                                  ;;******************************************************************
002483                                  .ENABL  LSB
002484  010166  000000                $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
002485  010170  000000                $TKQIN: .WORD   0               ;;INPUT POINTER
002486  010172  000000                $TKQOUT: .WORD  0               ;;OUTPUT POINTER
002487  010174  000007                $TKQSRT: .BLKB  7               ;;TTY KEYBOARD QUEUE
002488          010203                $TKQEND=.
002489          010204                .EVEN
002490
002491                                  ;*TK INITIALIZE ROUTINE
002492                                  ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
002493                                  ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
002494                                  ;
002495                                  ;*CALL:
```

```
                                        ;*      JSR     PC,$TKINT
                                        ;*      RETURN
                                        ;
010204  005037  010166                  $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
010210  012737  010174  010170                  MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
010216  013737  010170  010172                  MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
010224  012737  010254  000060                  MOV     #$TKSRV,@#^KVEC ;;INITIALIZE THE KEYBOARD VECTOR
010232  012737  000200  000062                  MOV     #200,@#^KVEC+2  ;;"BR" LEVEL 4
010240  005777  170702                          TST     @$TKB           ;;CLEAR DONE FLAG
010244  012777  000100  170672                  MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
010252  000207                                  RTS     PC              ;;RETURN TO CALLER
                                        ;
                                        ;*TK SERVICE ROUTINE
                                        ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
                                        ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
                                        ;*IT IN THE QUEUE.
                                        ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
                                        ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (@CNTLC)
                                        ;
010254  117746  170666                  $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
010260  042716  177600                          BIC     #^C177,(SP)     ;;STRIP THE JUNK
010264  021627  000003                          CMP     (SP),#3         ;;IS IT A CONTROL C?
010270  001007                                  BNE     1$              ;;BRANCH IF NO
010272  104401  011367                          TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
010276  004737  010204                          JSR     PC,$TKINT       ;;INIT THE KEYBOARD
010302  005726                                  TST     (SP)+           ;;CLEAN UP STACK
010304  000177  170770                          JMP     @CNTLC          ;;CONTROL C RESTART
010310  021627  000007                  1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
010314  001004                                  BNE     2$              ;;BRANCH IF NO
010316  022737  000176  001140                  CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
010324  001500                                  BEQ     6$              ;;GO TO SWR CHANGE
                                        ;
010326                                  2$:
010326  022737  000007  010166                  CMP     #7,$TKCNT       ;;IS THE QUEUE FULL?
010334  001004                                  BNE     3$              ;;BRANCH IF NO
010336  104401  001162                          TYPE    ,$BELL          ;;RING THE TTY BELL
010342  005726                                  TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
010344  000451                                  BR      5$              ;;EXIT
010346  021627  000023                  3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
010352  001021                                  BNE     32$             ;;BRANCH IF NO
010354  005077  170564                          CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
010360  005726                                  TST     (SP)+           ;;CLEAN CHAR OFF STACK
010362  105777  170556                  31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
010366  100375                                  BPL     31$             ;;LOOP UNTIL ITS THERE
010370  117746  170552                          MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
010374  042716  177600                          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
010400  022627  000021                          CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
010404  001366                                  BNE     31$             ;;BRANCH IF NO
010406  012777  000100  170530                  MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
010414  000002                                  RTI                     ;;RETURN
010416  005237  010166                  32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
010422  021627  000140                          CMP     (SP),#140       ;;IS IT UPPER CASE?
010426  002405                                  BLT     4$              ;;BRANCH IF YES
010430  021627  000175                          CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
010434  003002                                  BGT     4$              ;;BRANCH IF YES
010436  042716  000040                          BIC     #40,(SP)        ;;MAKE IT UPPER CASE
```

```
                                4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
                                        INC     $TKQIN          ;;UPDATE THE POINTER
                                        CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
                                        BNE     5$              ;;BRANCH IF NO
                                        MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
                                5$:     RTI                     ;;RETURN

                                ;;*****************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
                                ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
                                $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
                                        BNE     15$             ;;EXIT IF NOT
                                        TSTB    @$TKS           ;;IS A CHAR WAITING?
                                        BPL     15$             ;;IF NOT, EXIT
                                        MOVB    @$TKB,-(SP)     ;;YES
                                        BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
                                        CMP     (SP),#7         ;;IS IT A CONTROL-G?
                                        BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
                                                                ;;AND EXIT

                                ;;*****************************************************************
                                ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
                                ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
                                ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
                                6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
                                        BEQ     2$              ;;BRANCH IF YES
                                        TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
                                        JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
                                        CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
                                        MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

                                        TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
                                $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
                                        MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
                                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                        TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
                                19$:    CLR     -(SP)           ;;CLEAR COUNTER
                                        CLR     -(SP)           ;;THE NEW SWR
                                7$:     TSTB    @$TKS           ;;CHAR THERE?
                                        BPL     7$              ;;IF NOT TRY AGAIN

                                        MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
                                        BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

                                        CMP     (SP),#3         ;;IS IT A CONTROL-C?
                                        BNE     9$              ;;BRANCH IF NOT
                                        TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
                                        ADD     #6,SP           ;;CLEAN UP STACK
                                        CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
                                        BNE     8$              ;;BRANCH IF NO
                                        MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
                                8$:     JMP     @CNTLC          ;;CONTROL-C RESTART
```

```
2608   010662   021627   000025            9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
2609   010666   001005                             BNE     10$             ;;BRANCH IF NOT
2610   010670   104401   011374                    TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (↑U)
2611   010674   062706   000006            20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
2612   010700   000737                             BR      19$             ;;LET'S TRY IT AGAIN
2613
2614
2615   010702   021627   000015            10$:    CMP     (SP),#15        ;;IS IT A <CR>?
2616   010706   001022                             BNE     16$             ;;BRANCH IF NO
2617   010710   005766   000004                    TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
2618   010714   001403                             BEQ     11$             ;;BRANCH IF YES
2619   010716   016677   000002   170214           MOV     2(SP),@SWR      ;;SAVE NEW SWR
2620   010724   062706   000006            11$:    ADD     #6,SP           ;;CLEAR UP STACK
2621   010730   104401   001167            14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
2622   010734   123727   001135   000001           CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
2623   010742   001003                             BNE     15$             ;;BRANCH IF NOT
2624   010744   012777   000100   170172           MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
2625   010752   000002            15$:    RTI                             ;;RETURN
2626   010754   004737   007444            16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
2627   010760   021627   000060                    CMP     (SP),#60        ;;CHAR < 0?
2628   010764   002420                             BLT     18$             ;;BRANCH IF YES
2629   010766   021627   000067                    CMP     (SP),#67        ;;CHAR > 7?
2630   010772   003015                             BGT     18$             ;;BRANCH IF YES
2631   010774   042726   000060                    BIC     #60,(SP)+       ;;STRIP-OFF ASCII
2632   011000   005766   000002                    TST     2(SP)           ;;IS THIS THE FIRST CHAR
2633   011004   001403                             BEQ     17$             ;;BRANCH IF YES
2634   011006   006316                             ASL     (SP)            ;;NO, SHIFT PRESENT
2635   011010   006316                             ASL     (SP)            ;;   CHAR OVER TO MAKE
2636   011012   006316                             ASL     (SP)            ;;   ROOM FOR NEW ONE.
2637   011014   005266   000002            17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
2638   011020   056616   177776                    BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
2639   011024   000667                             BR      7$              ;;GET THE NEXT ONE
2640   011026   104401   001166            18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
2641   011032   000720                             BR      20$             ;;SIMULATE CONTROL-U
2642                                      .DSABL  LSB
2643
2644
2645                                      ;;**************************************************************
2646                                      ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2647                                      ;*CALL:
2648                                      ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
2649                                      ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
2650                                      ;*                              ;;WITH PARITY BIT STRIPPED OFF
2651                                      ;
2652
2653   011034   011646            $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
2654   011036   016666   000004   000002          MOV     4(SP),2(SP)     ;;THE PS
2655   011044   005066   000004                   CLR     4(SP)           ;;GET READY FOR A CHARACTER
2656   011050   005046                            CLR     -(SP)           ;;PUT NEW PS ON STACK
2657   011052   012746   011060                   MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
2658   011056   000002                            RTI                     ;;POP NEW PC AND PS
2659   011060                            64$:
2660   011060   005737   010166            1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
2661   011064   001775                            BEQ     1$
2662   011066   005337   010166                   DEC     $TKCNT          ;;DECREMENT THE COUNTER
2663   011072   117766   177074   000004          MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
```

```
2664  011100  005237  010172          INC     $TKQOUT          ;;UPDATE THE POINTER
2665  011104  023727  010172  010203  CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
2666  011112  001003                  BNE     2$               ;;BRANCH IF NO
2667  011114  012737  010174  010172  MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
2668  011122  000002          2$:     RTI                      ;;RETURN
2669                          ;;*********************************************************
2670                          ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2671                          ;*CALL:
2672                          ;*      RDLIN                    ;;INPUT A STRING FROM THE TTY
2673                          ;*      RETURN HERE              ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2674                          ;*                               ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
2675
2676  011124  010346          $RDLIN: MOV     R3,-(SP)         ;;SAVE R3
2677  011126  005046                  CLR     -(SP)            ;;CLEAR THE RUBOUT KEY
2678  011130  012703  011360  1$:     MOV     #$TTYIN,R3       ;;GET ADDRESS
2679  011134  022703  011367  2$:     CMP     #$TTYIN+7,R3     ;;BUFFER FULL?
2680  011140  101456                  BLOS    4$               ;;BR IF YES
2681  011142  104410                  RDCHR                    ;;GO READ ONE CHARACTER FROM THE TTY
2682  011144  112613                  MOVB    (SP)+,(R3)       ;;GET CHARACTER
2683  011146  122713  000177  10$:    CMPB    #177,(R3)        ;;IS IT A RUBOUT
2684  011152  001022                  BNE     5$               ;;BR IF NO
2685  011154  005716                  TST     (SP)             ;;IS THIS THE FIRST RUBOUT?
2686  011156  001007                  BNE     6$               ;;BR IF NO
2687  011160  112737  000134  011356  MOVB    #'\,9$           ;;TYPE A BACK SLASH
2688  011166  104401  011356          TYPE    ,9$
2689  011172  012716  177777          MOV     #-1,(SP)         ;;SET THE RUBOUT KEY
2690  011176  005303          6$:     DEC     R3               ;;BACKUP BY ONE
2691  011200  020327  011360          CMP     R3,#$TTYIN       ;;STACK EMPTY?
2692  011204  103434                  BLO     4$               ;;BR IF YES
2693  011206  111337  011356          MOVB    (R3),9$          ;;SETUP TO TYPEOUT THE DELETED CHAR.
2694  011212  104401  011356          TYPE    ,9$              ;;GO TYPE
2695  011216  000746                  BR      2$               ;;GO READ ANOTHER CHAR.
2696  011220  005716          5$:     TST     (SP)             ;;RUBOUT KEY SET?
2697  011222  001406                  BEQ     7$               ;;BR IF NO
2698  011224  112737  000134  011356  MOVB    #'\,9$           ;;TYPE A BACK SLASH
2699  011232  104401  011356          TYPE    ,9$
2700  011236  005016                  CLR     (SP)             ;;CLEAR THE RUBOUT KEY
2701  011240  122713  000025  7$:     CMPB    #25,(R3)         ;;IS CHARACTER A CTRL U?
2702  011244  001003                  BNE     8$               ;;BR IF NO
2703  011246  104401  011374          TYPE    ,$CNTLU          ;;TYPE A CONTROL "U"
2704  011252  000726                  BR      1$               ;;GO START OVER
2705  011254  122713  000022  8$:     CMPB    #22,(R3)         ;;IS CHARACTER A "↑R"?
2706  011260  001011                  BNE     3$               ;;BRANCH IF NO
2707  011262  105013                  CLRB    (R3)             ;;CLEAR THE CHARACTER
2708  011264  104401  001167          TYPE    ,$CRLF           ;;TYPE A "CR" & "LF"
2709  011270  104401  011360          TYPE    ,$TTYIN          ;;TYPE THE INPUT STRING
2710  011274  000717                  BR      2$               ;;GO PICKUP ANOTHER CHACTER
2711  011276  104401  001166  4$:     TYPE    ,$QUES           ;;TYPE A '?'
2712  011302  000712                  BR      1$               ;;CLEAR THE BUFFER AND LOOP
2713  011304  111337  011356  3$:     MOVB    (R3),9$          ;;ECHO THE CHARACTER
2714  011310  104401  011356          TYPE    ,9$
2715  011314  122723  000015          CMPB    #15,(R3)+        ;;CHECK FOR RETURN
2716  011320  001305                  BNE     2$               ;;LOOP IF NOT RETURN
2717  011322  105063  177777          CLRB    -1(R3)           ;;CLEAR RETURN (THE 15)
2718  011326  104401  001170          TYPE    ,$LF             ;;TYPE A LINE FEED
2719  011332  005726                  TST     (SP)+            ;;CLEAN RUBOUT KEY FROM THE STACK
```

```
2720  011334  012603                          MOV     (SP)+,R3        ;;RESTORE R3
2721  011336  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2722  011340  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
2723  011346  012766  011360  000004          MOV     #$TTYIN,4(SP)
2724  011354  000002                          RTI                     ;;RETURN
2725  011356     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
2726  011357     000                          .BYTE   0               ;;TERMINATOR
2727  011360  000007                  $TTYIN: .BLKB   7               ;;RESERVE 7 BYTES FOR TTY INPUT
2728  011367     136  006503  000012  $CNTLC: .ASCIZ  /↑C/<15><12>    ;;CONTROL "C"
2729  011374  052536  005015     000  $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
2730  011401     136  006507  000012  $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
2731  011406  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
2732  011414  020075     000
2733  011417     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
2734  011424  036440  000040
2735
2736                                  .SBTTL  TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
2737
2738                                  ;;*****************************************************************
2739                                  ;*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
2740                                  ;*LEADING NUMBERS.
2741                                  ;*CALL
2742                                  ;*      MOV     #NUMADR,-(SP)   ;;FIRST ADDRESS OF ASCIZ STRING
2743                                  ;*      JSR     PC,@#$SUPRS
2744
2745
2746  011430  010046                  $SUPRS: MOV     R0,-(SP)        ;;SAVE R0
2747  011432  016600  000004                  MOV     4(SP),R0        ;;PICKUP THE POINTER
2748  011436  105710                  1$:     TSTB    (R0)            ;;TERMINATEOR?
2749  011440  001403                          BEQ     2$              ;;BR IF YES
2750  011442  122720  000060                  CMPB    #'0,(R0)+       ;;IS THIS AN ASCII "0" ?
2751  011446  001773                          BEQ     1$              ;;BR IF YES
2752  011450  005300                  2$:     DEC     R0              ;;BACKUP BY "1"
2753  011452  010037  011460                  MOV     R0,3$           ;;SAVE FOR TYPING
2754  011456  104401                          TYPE                    ;;GO TYPE
2755  011460  000000                  3$:     .WORD   0               ;;ASCIZ POINTER GOES HERE
2756  011462  012600                          MOV     (SP)+,R0        ;;RESTORE R0
2757  011464  012616                          MOV     (SP)+,(SP)      ;;RESTORE THE STACK
2758  011466  000207                          RTS     PC              ;;RETURN
2759
2760                                  .SBTTL  SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
2761
2762                                  ;;*****************************************************************
2763                                  ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
2764                                  ;*UNSIGNED DECIMAL ASCIZ NUMBER.
2765                                  ;*CALL
2766                                  ;*      MOV     NUMBER,-(SP)    ;;PUT BINARY NUMBER ON THE STACK
2767                                  ;*      JSR     PC,@#$SB2D      ;;CALL
2768                                  ;*      RETURN                  ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK
2769
2770
2771  011470  016637  000002  011520  $SB2D:  MOV     2(SP),1$        ;;SAVE BINARY NUMBER
2772  011476  012746  011520                  MOV     #1$,-(SP)       ;;SET POINTER
2773  011502  004737  011524                  JSR     PC,@#$DB2D      ;;CALL DOUBLE LENGTH CONVERT
2774  011506  062716  000005                  ADD     #5,(SP)         ;;ONLY ALLOW FIVE CHARACTERS
2775  011512  012666  000002                  MOV     (SP)+,2(SP)     ;;PICKUP POINTER
```

```
                                        RTS     PC                      ;;RETURN
011520  000000  000000        1$:     .WORD   0,0

                                .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

                                ;;**************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
                                ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
                                ;*POSITIVE.
                                ;*CALL
                                ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      JSR     PC,@#$DB2D
                                ;*      RETURN                  ;;THE FIRST ADDRESS OF ASCIZ
                                                                ;;IS ON THE STACK

011524  104412          $DB2D:  SAVREG                          ;;SAVE REGISTERS
011526  016602  000002          MOV     2(SP),R2                ;;PICKUP THE DATA POINTER
011532  012700  011704          MOV     #$DECVL,R0              ;;GET ADDRESS OF "$DECVL" STRING
011536  010066  000002          MOV     R0,2(SP)                ;;PUT ADDRESS OF ASCIZ STRING ON STACK
011542  012201                  MOV     (R2)+,R1                ;;PICKUP THE BINARY NUMBER
011544  012202                  MOV     (R2)+,R2
011546  012737  000012  011622  MOV     #10.,4$                 ;;SET UP TO DO 10 CONVERSIONS
011554  012704  011634          MOV     #$TNPWR,R4              ;;ADDRESS OF TEN POWER
011560  012705  011636          MOV     #$TNPWR+2,R5
011564  005003          1$:     CLR     R3                      ;;CLEAR PARTIAL
011566  161401          2$:     SUB     (R4),R1                 ;;SUBTRACT TEN POWER
011570  005602                  SBC     R2
011572  161502                  SUB     (R5),R2
011574  002402                  BLT     3$                      ;;BR IF TEN POWER TO LARGE
011576  005203                  INC     R3                      ;;ADD 1 TO PARTIAL
011600  000772                  BR      2$                      ;;LOOP
011602  062401          3$:     ADD     (R4)+,R1                ;;RESTORE SUBTRACTED VALUE
011604  005502                  ADC     R2
011606  062402                  ADD     (R4)+,R2
011610  022525                  CMP     (R5)+,(R5)+             ;;MOVE TO NEXT TEN POWER
011612  052703  000060          BIS     #'0,R3                  ;;CHANGE PARTIAL TO ASCII
011616  110320                  MOVB    R3,(R0)+                ;;SAVE IT
011620  005327                  DEC     (PC)+                   ;;DONE?
011622  000000          4$:     .WORD   0
011624  001357                  BNE     1$                      ;;BR IF NO
011626  105020                  CLRB    (R0)+                   ;;TERMINATOR
011630  104413                  RESREG                          ;;RESTORE REGISTERS
011632  000207                  RTS     PC                      ;;RETURN
011634  145000          $TNPWR: 145000                          ;;1.0E09
011636  035632                  35632
011640  160400                  160400                          ;;1.0E08
011642  002765                  2765
011644  113200                  113200                          ;;1.0E07
011646  000230                  230
011650  041100                  041100                          ;;1.0E06
011652  000017                  17
011654  103240                  103240                          ;;1.0E05
011656  000001                  1
011660  023420                  23420                           ;;1.0E04
011662  000000                  0
```

```
2832  011664  001750                        1750                    ;;1.0E03
2833  011666  000000                        0
2834  011670  000144                        144                     ;;1.0E02
2835  011672  000000                        0
2836  011674  000012                        12                      ;;1.0E01
2837  011676  000000                        0
2838  011700  000001                        1                       ;;1.0E00
2839  011702  000000                        0
2840  011704  000014              $DECVL:   .BLKB   12.             ;;RESERVE STORAGE FOR ASCIZ STRING
2841
2842                              .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
2843
2844                              ;;*****************************************************************
2845                              ;*SAVE R0-R5
2846                              ;*CALL:
2847                              ;*      SAVREG
2848                              ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
2849                              ;*
2850                              ;*TOP---(+16)
2851                              ;* +2---(+18)
2852                              ;* +4---R5
2853                              ;* +6---R4
2854                              ;* +8---R3
2855                              ;*+10---R2
2856                              ;*+12---R1
2857                              ;*+14---R0
2858
2859  011720                      $SAVREG:
2860  011720  010046                      MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2861  011722  010146                      MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2862  011724  010246                      MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2863  011726  010346                      MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2864  011730  010446                      MOV     R4,-(SP)        ;;PUSH R4 ON STACK
2865  011732  010546                      MOV     R5,-(SP)        ;;PUSH R5 ON STACK
2866  011734  016646  000022              MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
2867  011740  016646  000022              MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
2868  011744  016646  000022              MOV     24(SP),-(SP)    ;;SAVE PS OF CALL
2869  011750  016646  000022              MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
2870  011754  000002                      RTI
2871
2872                              ;*RESTORE R0-R5
2873                              ;*CALL:
2874                              ;*      RESREG
2875  011756                      $RESREG:
2876  011756  012666  000022              MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
2877  011762  012666  000022              MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
2878  011766  012666  000022              MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
2879  011772  012666  000022              MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
2880  011776  012605                      MOV     (SP)+,R5        ;;POP STACK INTO R5
2881  012000  012604                      MOV     (SP)+,R4        ;;POP STACK INTO R4
2882  012002  012603                      MOV     (SP)+,R3        ;;POP STACK INTO R3
2883  012004  012602                      MOV     (SP)+,R2        ;;POP STACK INTO R2
2884  012006  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
2885  012010  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
2886  012012  000002                      RTI
2887
```

```
                                        .SBTTL  TRAP DECODER

                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                        ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                        ;*GO TO THAT ROUTINE.

012014  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
012016  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
012022  005740                          TST     -(R0)           ;;BACKUP BY 2
012024  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
012026  006300                          ASL     R0              ;;POSITION FOR INDEXING
012030  016000  012050                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
012034  000200                          RTS     R0              ;;GO TO ROUTINE

                                        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

012036  011646                 $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
012040  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
012046  000002                          RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

                                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                        ;*BY THE "TRAP" INSTRUCTION.

                                        ;       ROUTINE
                                        ;       -------
012050  012036                 $TRPAD: .WORD   $TRAP2
012052  007274                         $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
012054  007540                         $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
012056  007514                         $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
012060  007554                         $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
012062  007742                         $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

012064  010562                         $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

012066  010472                         $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
012070  011034                         $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
012072  011124                         $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
012074  011720                         $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
012076  011756                         $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE


                                .SBTTL  SINGLE/DUAL PORT RH11/RP04/5/6 DRIVER (REV 1.0)

                                ;COPYRIGHT (C) 1976
                                ;DIGITAL EQUIPMENT CORP.
                                ;MAYNARD, MA 01754
                                ;AUTHOR(S): JIM LACEY/CHUCK HESS

                                ;;*****************************************************************
```

```
                                        ;STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR "2"
                                                ;RPERRS   = RPDS1
                                                ;RPERRS+2 = RPER1
                                                ;RPERRS+4 = RPER2
                                                ;RPERRS+6 = RPER3
2950  012100  000000  000000  000000  RPERRS: .WORD   0,0,0,0
2951  012106  000000
                                        ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
                                                ;DRVACT=0 IF DRIVE IS IDLE
                                                ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
                                                ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
2958  012110      000               DRVACT: .BYTE   0               ;DRIVE 0
2959  012111      000                       .BYTE   0               ;DRIVE 1
2960  012112      000                       .BYTE   0               ;DRIVE 2
2961  012113      000                       .BYTE   0               ;DRIVE 3
2962  012114      000                       .BYTE   0               ;DRIVE 4
2963  012115      000                       .BYTE   0               ;DRIVE 5
2964  012116      000                       .BYTE   0               ;DRIVE 6
2965  012117      000                       .BYTE   0               ;DRIVE 7
                                        ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
                                                ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXSITENT
                                                ;DRVSTA>0 IF DRIVE IS ONLINE
                                                ;DRVSTA<0 IF DRIVE IS UNSAFE
2972  012120      000               DRVSTA: .BYTE   0               ;DRIVE 0
2973  012121      000                       .BYTE   0               ;DRIVE 1
2974  012122      000                       .BYTE   0               ;DRIVE 2
2975  012123      000                       .BYTE   0               ;DRIVE 3
2976  012124      000                       .BYTE   0               ;DRIVE 4
2977  012125      000                       .BYTE   0               ;DRIVE 5
2978  012126      000                       .BYTE   0               ;DRIVE 6
2979  012127      000                       .BYTE   0               ;DRIVE 7
                                        ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
                                                ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
                                                ;DRVTYP=1 IF DRIVE IS RP04
                                                ;DRVTYP=2 IF DRIVE IS RP05
                                                ;DRVTYP=4 IF DRIVE IS RP06
                                                ;DRVTYP=-1 IF NOT RP04/5/6
2988  012130      000               DRVTYP: .BYTE   0               ;DRIVE 0
2989  012131      000                       .BYTE   0               ;DRIVE 1
2990  012132      000                       .BYTE   0               ;DRIVE 2
2991  012133      000                       .BYTE   0               ;DRIVE 3
2992  012134      000                       .BYTE   0               ;DRIVE 4
2993  012135      000                       .BYTE   0               ;DRIVE 5
2994  012136      000                       .BYTE   0               ;DRIVE 6
2995  012137      000                       .BYTE   0               ;DRIVE 7
                                        ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
                                                ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
                                                ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
```

```
3000
3001  012140  000                 DPINT:  .BYTE   0               ;DRIVE 0
3002  012141  000                         .BYTE   0               ;DRIVE 1
3003  012142  000                         .BYTE   0               ;DRIVE 2
3004  012143  000                         .BYTE   0               ;DRIVE 3
3005  012144  000                         .BYTE   0               ;DRIVE 4
3006  012145  000                         .BYTE   0               ;DRIVE 5
3007  012146  000                         .BYTE   0               ;DRIVE 6
3008  012147  000                         .BYTE   0               ;DRIVE 7
3009
3010                              ;TABLE OF PENDING DUAL PORT REQUESTS
3011                                      ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
3012                                      ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
3013
3014  012150  000                 DPRQS:  .BYTE   0               ;DRIVE 0
3015  012151  000                         .BYTE   0               ;DRIVE 1
3016  012152  000                         .BYTE   0               ;DRIVE 2
3017  012153  000                         .BYTE   0               ;DRIVE 3
3018  012154  000                         .BYTE   0               ;DRIVE 4
3019  012155  000                         .BYTE   0               ;DRIVE 5
3020  012156  000                         .BYTE   0               ;DRIVE 6
3021  012157  000                         .BYTE   0               ;DRIVE 7
3022
3023                              ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
3024                                      ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
3025                                      ;"DPB" OF THE I/O OPERATION.
3026
3027  012160  000000              TRNSWT: .WORD   0
3028
3029                              ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
3030                                      ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
3031                                      ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
3032                                      ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
3033                                      ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
3034
3035  012162  000000              SRCHWT: .WORD   0
3036
3037                              ;RP04/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
3038                                      ;ACTDRV=0 IF DRIVER IS INACTIVE
3039                                      ;ACTDRV>0 IF DRIVER IS ACTIVE
3040
3041  012164  000                 ACTDRV: .BYTE   0
3042
3043                              ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
3044                                      ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
3045                                      ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
3046
3047  012165  000                 ACTSTR: .BYTE   0
3048
3049                              ;UNLOAD FLAG (ULDFLG=8 BYTES)
3050                                      ;ULDFLG=0 IF NO UNLOAD COMMAND
3051                                      ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
3052                                      ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
3053
3054  012166  000                 ULDFLG: .BYTE   0               ;DRIVE 0
3055  012167  000                         .BYTE   0               ;DRIVE 1
```

```
3056  012170  000                     .BYTE   0               ;DRIVE 2
3057  012171  000                     .BYTE   0               ;DRIVE 3
3058  012172  000                     .BYTE   0               ;DRIVE 4
3059  012173  000                     .BYTE   0               ;DRIVE 5
3060  012174  000                     .BYTE   0               ;DRIVE 6
3061  012175  000                     .BYTE   0               ;DRIVE 7
3062
3063                          ;LOOK AHEAD COUNT (LACNT=8 BYTES)
3064                                  ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
3065
3066  012176  000             LACNT:  .BYTE   0               ;DRIVE 0
3067  012177  000                     .BYTE   0               ;DRIVE 1
3068  012200  000                     .BYTE   0               ;DRIVE 2
3069  012201  000                     .BYTE   0               ;DRIVE 3
3070  012202  000                     .BYTE   0               ;DRIVE 4
3071  012203  000                     .BYTE   0               ;DRIVE 5
3072  012204  000                     .BYTE   0               ;DRIVE 6
3073  012205  000                     .BYTE   0               ;DRIVE 7
3074
3075                          ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
3076                                  ;SAVEFG <0 IF SAVE THE RH11/RP04/5/6 REGISTERS WHEN THE
3077                                  ;OPERATION IS COMPLETED AS PER (DPB+14).
3078                                  ;SAVEFG=0 IF SAVE THE RH11/RP04/5/6 REGISTERS, AS PER
3079                                  ;(DPB+14), AFTER AN ERROR.
3080
3081  012206  000000          SAVEFG: .WORD   0
3082
3083                          ;SEEK FLAG (SEEKFG=1 WORD)
3084                                  ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
3085                                  ;FOR A DATA TRANSFER START A SEARCH COMMAND
3086                                  ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
3087                                  ;DISREGARD THE WINDOW
3088
3089  012210  000000          SEEKFG: .WORD   0
3090
3091                          ;TIMEOUT TABLE (TIMER=8 WORDS)
3092                                  ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
3093
3094  012212  177777          TIMER:  .WORD   -1              ;DRIVE 0
3095  012214  177777                  .WORD   -1              ;DRIVE 1
3096  012216  177777                  .WORD   -1              ;DRIVE 2
3097  012220  177777                  .WORD   -1              ;DRIVE 3
3098  012222  177777                  .WORD   -1              ;DRIVE 4
3099  012224  177777                  .WORD   -1              ;DRIVE 5
3100  012226  177777                  .WORD   -1              ;DRIVE 6
3101  012230  177777                  .WORD   -1              ;DRIVE 7
3102
3103                          ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
3104                                  ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
3105                                  ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
3106
3107  012232  177777          DTUW:   .WORD   -1
3108
3109                          ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
3110                                  ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
3111                                  ;ATTENTION BIT
```

```
                                ATABIT: .BYTE   1        ;DRIVE 0
                                        .BYTE   2        ;DRIVE 1
                                        .BYTE   4        ;DRIVE 2
                                        .BYTE   10       ;DRIVE 3
                                        .BYTE   20       ;DRIVE 4
                                        .BYTE   40       ;DRIVE 5
                                        .BYTE   100      ;DRIVE 6
                                        .BYTE   200      ;DRIVE 7

                                ;RP04/5/6 TO RH11 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
                                ;CALLING IT FATAL (MCPEMX=1 WORD)

012244  000003                  MCPEMX: .WORD   3

                                ;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RP04/5/6),
                                ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).

012246  176700                  RPADR:  .WORD   176700
012250  000254  000240          RPVEC:  .WORD   254,5*32.

                                ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)

012254  000004                  MXLACT: .WORD   4
                                ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)

012256  001000                  MXDLTA: .WORD   8.*64.
                                ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)

012260  000200                  MNDLTA: .WORD   2*64.
                                ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)

012262  000005                  MXWNDW: .WORD   5

                                ;DEFINITION OF THE RH11/RP04/5/6 ADDRESS INDEXES

        000000                  RPCS1=0                 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
        000002                  RPWC=2                  ;WORD COUNT REGISTER (NOT A DRIVE REG)
        000004                  RPBA=4                  ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
        000006                  RPDA=6                  ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
        000010                  RPCS2=10                ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
        000012                  RPDS1=12                ;DRIVE STATUS REGISTER (DRIVE REG 01)
        000014                  RPER1=14                ;ERROR REGISTER #1 (DRIVE REG. 02)
        000016                  RPAS=16                 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
        000020                  RPLA=20                 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
        000022                  RPDB=22                 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
        000024                  RPMR=24                 ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
        000026                  RPDT=26                 ;DRIVE TYPE REGISTER (DRIVE REG. 06)
        000030                  RPSN=30                 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
        000032                  RPOF=32                 ;OFFSET REGISTER (DRIVE REG. 11)
        000034                  RPCA=34                 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
        000036                  RPCC=36                 ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
        000040                  RPER2=40                ;ERROR REGISTER #2 (DRIVE REG. 14)
        000042                  RPER3=42                ;ERROR REGISTER #3 (DRIVE REG. 15)
        000044                  RPEC1=44                ;ECC POSITION REGISTER (DRIVE REG. 16)
        000046                  RPEC2=46                ;ECC PATTERN REGISTER (DRIVE REG.  17)
```

```
                                        ;RH11/RP04/5/6 DRIVER INITIALIZATION CODE
                                                ;THIS ROUTINE WILL DETERMINE WHICH RP04/5/6 DRIVES ARE
                                                ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
                                                ;TO THE PROPER STATE FOR EACH DRIVE.
                                                ;NOTE: THIS ROUTINE CALLS DRVINT
                                        ;
                                        ;CALL
                                        ;
                                                JSR     PC,RPINIT
                                                RETURN
                                        ;
                                        ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
                                        ;
012264  104412                  RPINIT: SAVREG                          ;SAVE R0 - R5
012266  013746  177776                  MOV     @#PS,-(SP)              ;SAVE THE PRESENT PROCESSOR STATUS
012272  012737  000240  177776          MOV     #<5*32.>,@#PS           ;CHANGE THE PRIORITY TO 5
012300  004737  020250                  JSR     PC,CLRQUE               ;CLEAR ALL REQUEST QUEUES
012304  012701  012100                  MOV     #RPERRS,R1              ;FIRST ADDRESS TO BE CLEARED
012310  012702  012210                  MOV     #SEEKFG,R2              ;LAST ADDRESS TO BE CLEARED
012314  005021                  1$:     CLR     (R1)+                   ;CLEAR
012316  020102                          CMP     R1,R2                   ;ARE WE DONE?
012320  101775                          BLOS    1$                      ;BRANCH IF NO
012322  012702  012232                  MOV     #DTUW,R2                ;LAST ADDRESS
012326  012721  177777          2$:     MOV     #-1,(R1)+               ;INITIALIZE
012332  020102                          CMP     R1,R2                   ;DONE?
012334  101774                          BLOS    2$                      ;LOOP IF NO
012336  005037  012120                  CLR     DRVSTA                  ;SET ALL DRIVES TO OFFLINE
012342  005037  012122                  CLR     DRVSTA+2
012346  005037  012124                  CLR     DRVSTA+4
012352  005037  012126                  CLR     DRVSTA+6
012356  013703  012250                  MOV     RPVEC,R3                ;SETUP THE RH11/RP04/5/6 VECTOR
012362  012723  015130                  MOV     #ISR,(R3)+
012366  013713  012252                  MOV     RPVEC+2,(R3)
012372  013704  012246                  MOV     RPADR,R4                ;FIRST ADDRESS OF RH11/RP04
012376  012764  000040  000010          MOV     #BIT05,RPCS2(R4)         ;MASSBUS INIT
012404  005001                          CLR     R1                      ;START WITH DRIVE 0
012406  004037  012476          3$:     JSR     R0,DRVINT               ;INIT THE DRIVE
012412  000401                          BR      4$                      ;'DVA' NOT SET OR PARITY ERROR
012414  000402                          BR      5$                      ;NORMAL RETURN
012416  105061  012120          4$:     CLRB    DRVSTA(R1)              ;SET DRIVE STATUS TO OFFLINE
012422  005201                  5$:     INC     R1                      ;GO TO NEXT DRIVE
012424  042701  177770                  BIC     #^C7,R1                 ;MASK OUT UNUSED BITS
012430  001366                          BNE     3$                      ;BR IF MORE DRIVES TO GO
012432  012701  000007                  MOV     #7,R1                   ;START WITH DRIVE 7
012436  005037  177776                  CLR     @#PS                    ;CLEAR THE PROCESSOR STATUS
012442  105761  012140          6$:     TSTB    DPINT(R1)               ;WAITING FOR DRIVE TO SWITCH PORTS ?
012446  001405                          BEQ     8$                      ;BR NOT WAITING
012450  004737  017704                  JSR     PC,SET.IE               ;SET INTERRUPT
012454  105761  012140          7$:     TSTB    DPINT(R1)               ;DRIVE SWITCHED PORTS ?
012460  001375                          BNE     7$                      ;BR IF NOT
012462  005301                  8$:     DEC     R1                      ;GO TO THE NEXT DRIVE
012464  100366                          BPL     6$                      ;CHECK NEXT DRIVE
012466  012637  177776                  MOV     (SP)+,@#PS              ;RESTORE THE PROCESSOR STATUS
012472  104413                          RESREG                          ;RESTORE R0 - R5
012474  000207                          RTS     PC                      ;BYE-BYE
```

```
                                        ;DRIVE INITILIZATION ROUTINE
                                        ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
                                        ;AN RP04/5/6. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
                                        ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
                                        ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
                                        ;DRVSTA IS SET TO THE PROPER CONDITION.
                                        ;CALL
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                        ;       MOV     RPADR,R4        ;UNIBUS ADDRESS OF RH11/RP04/5/6 (RPCS1)
                                        ;       JSR     R0,DRVINT       ;CALLED BY A JSR
                                        ;       RETURN1                 ;ERROR OCCURRED (PARITY)
                                        ;       RETURN2                 ;NORMAL RETURN
                                        ;
012476  010546                  DRVINT: MOV     R5,-(SP)        ;SAVE R5
012500  105061  012120                  CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
012504  105061  012130                  CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
012510  105061  012166                  CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
012514  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT A DRIVE
012520  112764  000111  000000          MOVB    #111,RPCS1(R4)  ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
012526  032764  010000  000010          BIT     #BIT12,RPCS2(R4) ;NONEXISTENT DRIVE?
012534  001403                          BEQ     1$              ;NO---BRANCH
012536  004737  017704                  JSR     PC,SET.IE       ;GO SET "IE" WITHOUT A "TRE"
012542  000520                          BR      6$              ;LEAVE THIS ROUTINE
012544  105061  012120          1$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
012550  032764  004000  000000          BIT     #BIT11,RPCS1(R4)  ;SEE IF DRIVE AVAILABLE
012556  001514                          BEQ     7$              ;BR IF DRIVE NOT AVAILABLE
012560  004037  017214                  JSR     R0,RD.RP        ;READ THE DRIVE TYPE REG.
012564  000026                          RPDT
012566  013030                          8$                      ;ERROR RETURN ADDRESS
012570  012605                          MOV     (SP)+,R5        ;PUT DRIVE TYPE IN R5
012572  112761  000001  012130          MOVB    #1,DRVTYP(R1)   ;SET RP04 INDICATOR
012600  022705  020020                  CMP     #20020,R5       ;IS IT A SINGLE PORT RP04?
012604  001431                          BEQ     2$              ;BRANCH IF YES
012606  022705  024020                  CMP     #24020,R5       ;IS IT A DUAL PORT RP04?
012612  001426                          BEQ     2$              ;BR IF YES
012614  112761  000002  012130          MOVB    #2,DRVTYP(R1)   ;SET RP05 INDICATOR
012622  022705  020021                  CMP     #20021,R5       ;SINGLE PORT RP05 ?
012626  001420                          BEQ     2$              ;BR IF YES
012630  022705  024021                  CMP     #24021,R5       ;DUAL PORT RP05 ?
012634  001415                          BEQ     2$              ;BR IF YES
012636  112761  000004  012130          MOVB    #4,DRVTYP(R1)   ;SET RP06 INDICATOR
012644  022705  020022                  CMP     #20022,R5       ;SINGLE PORT RP06 ?
012650  001407                          BEQ     2$              ;BR IF YES
012652  022705  024022                  CMP     #24022,R5       ;DUAL PORT RP06 ?
012656  001404                          BEQ     2$              ;BR IF YES
012660  112761  177777  012130          MOVB    #-1,DRVTYP(R1)  ;SET INDICATOR TO 'OTHER'
012666  000446                          BR      6$              ;EXIT
012670  012746  000121          2$:     MOV     #121,-(SP)      ;DO A "READ-IN PRESET"
012674  004037  017374                  JSR     R0,WRT.RP
012700  000000                          RPCS1
012702  013030                          8$
012704  012746  010000                  MOV     #BIT12,-(SP)    ;SET FMT22=1
012710  004037  017374                  JSR     R0,WRT.RP
012714  000032                          RPOF
```

```
3280  012716  013030                        8$
3281  012720  004037  017214                JSR     R0,RD.RP        ;READ RPDS1
3282  012724  000012                        RPDS1
3283  012726  013030                        8$
3284  012730  012605                        MOV     (SP)+,R5        ;AND SAVE IT IN R5
3285  012732  100015                        BPL     4$              ;BRANCH IF ATA=0
3286  012734  116164  012234  000016        MOVB    ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
3287  012742  004037  017214                JSR     R0,RD.RP        ;FIND OUT WHY ATA=1
3288  012746  000014                        RPER1
3289  012750  013030                        8$
3290  012752  006126                        ROL     (SP)+           ;IS IT UNSAFE?
3291  012754  100004                        BPL     4$              ;BR IF NOT
3292  012756  112761  177777  012120        MOVB    #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
3293  012764  000407                        BR      6$              ;EXIT
3294  012766  005105                4$:     COM     R5              ;CHECK MOL, DPR, DRY, AND VV
3295  012770  042705  167077                BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
3296  012774  001003                        BNE     6$              ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
3297  012776  112761  000001  012120        MOVB    #1,DRVSTA(R1)   ;SET DRIVE STATUS TO ONLINE
3298  013004  005720                6$:     TST     (R0)+           ;STEP OVER THE ERROR RETURN
3299  013006  000410                        BR      8$              ;EXIT
3300  013010  006301                7$:     ASL     R1              ;CHANGE INDEX TO ADDRESS WORDS
3301  013012  012761  003720  012212        MOV     #2000.,TIMER(R1)  ;START 2 SEC TIMER
3302  013020  006201                        ASR     R1              ;RESTORE R1
3303  013022  105161  012140                COMB    DPINT(R1)       ;SET PORT INITIALIZE INIDICATOR
3304  013026  005720                        TST     (R0)+
3305  013030  012605                8$:     MOV     (SP)+,R5        ;RESTORE R5
3306  013032  000200                        RTS     R0              ;EXIT
3307
3308                                ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
3309                                ;
3310                                ;CALL
3311                                ;
3312                                ;       JSR     R0,@#RP04       ;CALL THE RP04/5/6 DRIVER
3313                                ;       PNTADR                  ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
3314                                ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
3315                                ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
3316                                                                ;IS AN ERROR CONDITION
3317
3318  013034  013746  177776        RP04:   MOV     @#PS,-(SP)      ;SAVE THE CALLING STATUS
3319  013040  013737  012252  177776        MOV     RPVEC+2,@#PS    ;DON'T ALLOW ANY RP04/5/6 INTERRUPTS
3320  013046  112737  000001  012164        MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
3321  013054  104412                        SAVREG                  ;SAVE R0 - R5
3322  013056  011002                        MOV     (R0),R2         ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
3323  013060  005062  000016                CLR     16(R2)          ;CLEAR THE STATUS/ERROR INDICATOR
3324  013064  111201                        MOVB    (R2),R1         ;PICKUP THE DRIVE NUMBER
3325  013066  013704  012246                MOV     RPADR,R4        ;UNIBUS ADDRESS OF RPCS1
3326  013072  105761  012120                TSTB    DRVSTA(R1)      ;CHECK DRIVES STATUS
3327  013076  003014                        BGT     1$              ;BRANCH IF ONLINE
3328  013100  105761  012166                TSTB    ULDFLG(R1)      ;UNLOAD COMMAND IN QUEUE?
3329  013104  001036                        BNE     3$              ;BRANCH IF YES
3330  013106  105761  012140                TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE
3331  013112  001042                        BNE     5$              ;BR IF YES
3332  013114  004037  012476                JSR     R0,DRVINT       ;GO INIT. THE DRIVE
3333  013120  000434                        BR      4$              ;ERROR RETURN
3334  013122  105761  012120                TSTB    DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
3335  013126  003445                        BLE     6$              ;BR IF NOT
```

```
3336  013130  105761  012150          1$:     TSTB    DPRQS(R1)           ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
3337  013134  001031                          BNE     5$                  ;BR IF YES
3338  013136  010164  000010                  MOV     R1,RPCS2(R4)        ;SELECT THE DRIVE
3339  013142  004037  020346                  JSR     R0,DRVQUE           ;PUT THIS REQUEST IN QUEUE
3340  013146  000460                          BR      9$                  ;QUEUE IS FULL
3341  013150  122762  000103  000002          CMPB    #103,2(R2)          ;IS THIS REQ. FOR AN UNLOAD?
3342  013156  001003                          BNE     2$                  ;BR IF NO
3343  013160  112761  177777  012166          MOVB    #-1,ULDFLG(R1)      ;SET THE "UNLOAD IN QUEUE" FLAG
3344  013166  105761  012110          2$:     TSTB    DRVACT(R1)          ;IS THIS DRIVE ACTIVE?
3345  013172  001043                          BNE     8$                  ;BR IF YES
3346  013174  004737  013326                  JSR     PC,OPT              ;CALL THE OPTIMIZER
3347  013200  000440                          BR      8$
3348  013202  012762  120000  000016  3$:     MOV     #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
3349  013210  000434                          BR      8$                  ;EXIT
3350  013212  004737  014436          4$:     JSR     PC,CI7              ;GO HANDLE THE PARITY ERROR
3351  013216  000431                          BR      8$
3352  013220  004037  020346          5$:     JSR     R0,DRVQUE           ;PUT REQUEST IN QUEUE
3353  013224  000431                          BR      9$                  ;QUEUE IS FULL
3354  013226  032714  000100                  BIT     #BIT06,(R4)         ;IS 'IE' SET ALREADY ?
3355  013232  001023                          BNE     8$                  ;BR IF IT IS
3356  013234  004737  017704                  JSR     PC,SET.IE           ;SET INTERRUPT
3357  013240  000420                          BR      8$                  ;RETURN, REQUEST IN QUEUE
3358  013242  105761  012120          6$:     TSTB    DRVSTA(R1)          ;SEE IF DRIVE OFFLINE OR UNSAFE
3359  013246  002412                          BLT     7$                  ;BR IF UNSAFE
3360  013250  012762  140000  000016          MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
3361  013256  105761  012130                  TSTB    DRVTYP(R1)          ;SEE IF OFFLINE OR NONEXISTENT
3362  013262  001007                          BNE     8$                  ;BR IF OFFLINE
3363  013264  012762  100002  000016          MOV     #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
3364  013272  000403                          BR      8$                  ;GO TO EXIT
3365  013274  012762  110000  000016  7$:     MOV     #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
3366  013302  104413                  8$:     RESREG                      ;RESTORE R0 - R5
3367  013304  005720                          TST     (R0)+               ;SETUP FOR NORMAL RETURN
3368  013306  000401                          BR      10$                 ;FINISH UP, THEN EXIT
3369  013310  104413                  9$:     RESREG                      ;RESTORE R0 - R5
3370  013312  005720                  10$:    TST     (R0)+               ;CORRECT THE RETURN ADDRESS
3371  013314  105037  012164                  CLRB    ACTDRV              ;CLEAR "ACTIVE DRIVER" FLAG
3372  013320  012637  177776                  MOV     (SP)+,@#PS          ;RETURN "PS" TO USER LEVEL
3373  013324  000200                          RTS     R0                  ;RETURN TO CALLER
3374
3375                                  ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
3376                                  ;
3377                                  ;CALL
3378                                  ;       MOV     #DRVNUM,R1          ;DRIVE NUMBER TO R1
3379                                  ;       JSR     PC,OPT              ;SETUP A COMMAND
3380                                  ;
3381  013326  104412                  OPT:    SAVREG                      ;SAVE R0 - R5
3382  013330  013746  177776                  MOV     @#PS,-(SP)          ;SAVE PROC. STATUS
3383  013334  146137  012234  012162          BICB    ATABIT(R1),SRCHWT           ;CLEAR "SEARCH WAIT" KEY
3384  013342  004737  020422                  JSR     PC,GETREQ           ;GET "DPB" POINTER OF REQUEST
3385  013346  005702                          TST     R2                  ;IS THERE A REQUEST IN QUEUE?
3386  013350  001505                          BEQ     7$                  ;NO--BRANCH TO EXIT
3387  013352  032764  004000  000000          BIT     #BIT11,RPCS1(R4)            ;IS DVA SET?
3388  013360  001407                          BEQ     10$                 ;BRANCH IF NOT
3389  013362  032764  000100  000012          BIT     #BIT6,RPDS1(R4)     ;IS VV SET ?
3390  013370  001003                          BNE     10$                 ;BR IF IT IS
3391  013372  004037  012476          9$:     JSR     R0,DRVINT           ;SEE IF DRIVE STILL ONLINE ?
```

```
                                        BR      6$                      ;PARITY OR 'DVA' NOT SET
                                10$:    TSTB    DRVSTA(R1)              ;IS DRIVE ONLINE?
                                        BGT     1$                      ;YES--BRANCH
                                        JSR     PC,POPQUE               ;NO--REMOVE REQUEST FROM QUEUE
                                        MOV     #BIT15!BIT14,16(R2)     ;SET OFFLINE STATUS/ERROR INDICATOR
                                        TSTB    DRVSTA(R1)              ;IS DRIVE UNSAFE ?
                                        BPL     8$                      ;BR TO EXIT IF NOT
                                        MOV     #BIT15!BIT12,16(R2)     ;SET UNSAFE STATUS/ERROR INDICATOR
                                        BR      8$                      ;BRANCH TO EXIT
                                1$:     MOV     #111,-(SP)              ;LOAD COMMAND ONTO THE STACK
                                        JSR     R0,WRT.RP               ;LOAD THE REGISTER
                                        RPCS1                           ;REGISTER INCREMENT
                                        6$                              ;ERROR RETURN ADDRESS
                                        BIT     #BIT11,(R4)             ;DRIVE AVAILABLE ?
                                        BEQ     5$                      ;BR IF NOT
                                        CMPB    #150,2(R2)              ;IS THE REQUEST FOR I/O?
                                        BLT     2$                      ;YES--BRANCH
                                        JSR     PC,CI4                  ;CALL THE COMMAND INITIATOR
                                        BR      8$                      ;BRANCH TO EXIT
                                2$:     TST     DTUW                    ;DATA TRANSFER UNDERWAY?
                                        BGE     4$                      ;YES--GO START A SEARCH
                                        TST     SEEKFG                  ;DO IMPLIED SEEKS?
                                        BMI     3$                      ;YES---BRANCH
                                        JSR     R0,LA                   ;NO--DO LOOK AHEAD
                                        BR      8$                      ;RETURN HERE ON A PARITY ERROR
                                        BR      4$                      ;GO START A SEARCH
                                3$:     JSR     PC,CI1                  ;START A DATA TRANSFER
                                        BR      8$
                                4$:     JSR     PC,CI3                  ;START A SEARCH
                                        BR      8$                      ;GO TO THE EXIT
                                5$:     MOVB    #-1,DPRQS(R1)           ;SET PORT REQUEST INDICATOR
                                        MOV     R1,R3                   ;SET UP TO ADDRESS WORDS
                                        ASL     R3                      ;CONVERT TO WORD INDEX
                                        MOV     #10000.,TIMER(R3)       ;START 10 SEC TIMER
                                        BR      7$                      ;EXIT
                                6$:     JSR     PC,CI7                  ;PROCESS THE PARITY ERROR
                                7$:     BIT     #BIT06,(R4)             ;SEE IF 'IE' ALREADY SET
                                        BNE     8$                      ;BR IF SET
                                        JSR     PC,SET.IE               ;SET "IE" WITHOUT A "TRE"
                                8$:     MOV     (SP)+,@#PS              ;RESTORE PROC. STATUS
                                        RESREG                          ;RESTORE R0 - R5
                                        RTS     PC

                                ;COMMAND INITIATOR
                                ;
                                ;CALL
                                ;       MOV     #DRVNUM,R1              ;DRIVE NUMBER
                                ;       MOV     #DPB,R2                 ;ADDRESS OF DPB
                                ;       JSR     PC,CI?                  ;CI?= CI1,CI3, OR CI4
                                ;                                       ;WHERE:
                                ;                                       ;CI1=DATA TRANSFER
                                ;                                       ;CI2=SEARCH REQUESTED BY DATA XFER
                                ;                                       ;CI4=NOT DATA TRANSFER

                                CI1:    JSR     PC,POPQUE               ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
                                        MOV     R2,TRNSWT               ;PUT REQ. IN TRANSFER WAIT QUEUE
```

```
3448  013616  010203                        MOV     R2,R3           ;DPB ADDRESS TO R3
3449  013620  013704  012246                MOV     RPADR,R4        ;RPCS1 ADDRESS
3450  013624  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
3451  013630  062703  000004                ADD     #4,R3           ;DESIRED WORD COUNT
3452  013634  062704  000002                ADD     #2,R4           ;RPWC ADDRESS
3453  013640  012324                        MOV     (R3)+,(R4)+     ;LOAD WORD COUNT
3454  013642  012324                        MOV     (R3)+,(R4)+     ;LOAD BUFFER ADDRESS
3455  013644  012346                        MOV     (R3)+,-(SP)     ;LOAD SECTOR AND TRACK
3456  013646  004037  017374                JSR     R0,WRT.RP       ;CALL THE LOAD(WRITE) ROUTINE
3457  013652  000006                        RPDA                    ;INDEX OF REGISTER TO LOAD
3458  013654  014436                        CI7                     ;ERROR RETURN ADDRESS
3459  013656  012346                        MOV     (R3)+,-(SP)     ;LOAD CYLINDER ADDRESS
3460  013660  004037  017374                JSR     R0,WRT.RP
3461  013664  000034                        RPCA
3462  013666  014436                        CI7
3463  013670  016246  000002                MOV     2(R2),-(SP)     ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
3464  013674  004037  017374                JSR     R0,WRT.RP
3465  013700  000000                        RPCS1
3466  013702  014436                        CI7
3467  013704  010137  012232                MOV     R1,DTUW         ;SET "DATA TRANSFER UNDERWAY"
3468  013710  000137  014400                JMP     CI5
3469  013714  013704  012246        CI3:    MOV     RPADR,R4        ;RPCS1 ADDRESS
3470  013720  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
3471  013724  016246  000012                MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
3472  013730  004037  017374                JSR     R0,WRT.RP
3473  013734  000034                        RPCA
3474  013736  014436                        CI7
3475  013740  116203  000010                MOVB    10(R2),R3       ;PICKUP SECTOR ADDRESS
3476  013744  163703  012262                SUB     MXWNDW,R3       ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
3477  013750  002002                        BGE     1$
3478  013752  062703  000026                ADD     #22.,R3
3479  013756  010346        1$:             MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
3480  013760  116266  000011  000001        MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
3481  013766  004037  017374                JSR     R0,WRT.RP       ;LOAD DESIRED TRACK & SECTOR
3482  013772  000006                        RPDA
3483  013774  014436                        CI7
3484  013776  012746  000131                MOV     #131,-(SP)      ;START A SEARCH
3485  014002  004037  017374                JSR     R0,WRT.RP
3486  014006  000000                        RPCS1
3487  014010  014436                        CI7
3488  014012  156137  012234  012162        BISB    ATABIT(R1),SRCHWT       ;SET "SEARCH WAIT" KEY
3489  014020  000567                        BR      CI5
3490  014022  013704  012246        CI4:    MOV     RPADR,R4        ;RPCS1 ADDRESS
3491  014026  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
3492  014032  116203  000002                MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
3493  014036  122703  000131                CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
3494  014042  001007                        BNE     1$              ;BRANCH IF NO
3495  014044  016246  000010                MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
3496  014050  004037  017374                JSR     R0,WRT.RP
3497  014054  000006                        RPDA
3498  014056  014436                        CI7
3499  014060  000403                        BR      2$              ;GO LOAD CYLINDER
3500  014062  122703  000105        1$:     CMPB    #105,R3         ;IS IT A SEEK COMMAND
3501  014066  001007                        BNE     3$              ;BRANCH IF NO
3502  014070  016246  000012        2$:     MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
3503  014074  004037  017374                JSR     R0,WRT.RP
```

```
3504  014100  000034                        RPCA
3505  014102  014436                        CI7
3506  014104  000546                        BR      CI6
3507  014106  122703  000115        3$:     CMPB    #115,R3         ;IS IT AN "OFFSET" COMMAND?
3508  014112  001013                        BNE     4$              ;BR IF NO
3509  014114  004037  017214                JSR     R0,RD.RP        ;MERGE THE OFFSET VALUE INTO RPOF
3510  014120  000032                        RPOF                    ;BUT DON'T CHANGE THE UPPER
3511  014122  014436                        CI7                     ;BYTE WHEN LOADING THE
3512  014124  116216  000001                MOVB    1(R2),(SP)      ;REGISTER (RPOF)
3513  014130  004037  017374                JSR     R0,WRT.RP
3514  014134  000032                        RPOF
3515  014136  014436                        CI7
3516  014140  000530                        BR      CI6             ;GO START THE COMMAND
3517  014142  122703  000107        4$:     CMPB    #107,R3         ;IS IT A "RECALIBRATE" COMMAND?
3518  014146  001525                        BEQ     CI6             ;BRANCH IF YES
3519  014150  122703  000117                CMPB    #117,R3         ;IS IT A RETURN TO CENTER?
3520  014154  001522                        BEQ     CI6             ;BRANCH IF YES
3521  014156  122703  000103                CMPB    #103,R3         ;IS IT AN "UNLOAD" COMMAND?
3522  014162  001016                        BNE     5$              ;BRANCH IF NO
3523  014164  112761  000001  012110        MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE INDICATOR
3524  014172  105061  012120                CLRB    DRVSTA(R1)      ;PUT DRIVE STATUS TO OFFLINE
3525  014176  112761  000001  012166        MOVB    #1,ULDFLG(R1)   ;SET "UNLOAD IN PROGRESS" FLAG
3526  014204  010346                        MOV     R3,-(SP)        ;START THE "UNLOAD" COMMAND
3527  014206  004037  017374                JSR     R0,WRT.RP
3528  014212  000000                        RPCS1
3529  014214  014436                        CI7
3530  014216  000207                        RTS     PC              ;RETURN TO USER
3531  014220  122703  000143        5$:     CMPB    #143,R3         ;IS IT A "SET FORMAT" COMMAND?
3532  014224  001014                        BNE     6$              ;BRANCH IF NO
3533  014226  004037  017214                JSR     R0,RD.RP        ;READ THE OFFSET REGISTER
3534  014232  000032                        RPOF
3535  014234  014436                        CI7
3536  014236  116266  000001  000001        MOVB    1(R2),1(SP)     ;COMBINE "FMT22", "ECI", AND "HCI"
3537  014244  004037  017374                JSR     R0,WRT.RP       ;LOAD "FMT22", "ECI", AND/OR "HCI".
3538  014250  000032                        RPOF
3539  014252  014436                        CI7
3540  014254  000436                        BR      12$
3541  014256  122703  000141        6$:     CMPB    #141,R3         ;IS IT A "GET REGISTER" COMMAND?
3542  014262  001023                        BNE     10$             ;BRANCH IF NO
3543  014264  016203  000006        7$:     MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
3544                                                                ;TO PUT THE REGISTER(S)
3545  014270  116237  000010  014306        MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
3546  014276  116205  000011                MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
3547  014302  004037  017214        8$:     JSR     R0,RD.RP        ;READ RP04/5/6 REGISTER
3548  014306  000000                9$:     RPCS1                   ;INDEX OF REG. TO READ
3549  014310  014436                        CI7
3550  014312  012623                        MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RH11/RP04/5/6 REG.
3551  014314  023705  014306                CMP     9$,R5           ;LAST REG. BEEN READ?
3552  014320  001414                        BEQ     12$             ;GET OUT IF YES
3553  014322  062737  000002  014306        ADD     #2,9$           ;INCREASE THE INDEX BY 2
3554  014330  000764                        BR      8$              ;LOOP--MORE TO READ
3555  014332  122703  000145        10$:    CMPB    #145,R3         ;IS IT A "SELECT DRIVE" COMMAND?
3556  014336  001405                        BEQ     12$             ;BRANCH IF YES
3557  014340  010346                11$:    MOV     R3,-(SP)        ;LOAD THE COMMAND
3558  014342  004037  017374                JSR     R0,WRT.RP
3559  014346  000000                        RPCS1
```

```
3560  014350  014436                        CI7
3561  014352  004737  020444          12$:  JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
3562  014356  052762  000200  000016        BIS     #BIT07,16(R2)   ;SET THE "DONE" BIT
3563  014364  005737  012206                TST     SAVEFG          ;SAVE THE RH11/RP04/5/6 REGISTERS?
3564  014370  100002                        BPL     13$             ;BRANCH IF NO
3565  014372  004737  017566                JSR     PC,SVRH11       ;YES--GO SAVE THE REGISTERS
3566  014376  000207                  13$:  RTS     PC              ;RETURN TO USER
3567  014400  006301                  CI5:  ASL     R1
3568  014402  012761  001750  012212        MOV     #1000.,TIMER(R1)        ;SET A ONE SECOND TIMER
3569  014410  006201                        ASR     R1
3570  014412  112761  000001  012110        MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
3571  014420  000207                        RTS     PC              ;RETURN TO THE USER
3572  014422  010346                  CI6:  MOV     R3,-(SP)        ;LOAD THE COMMAND
3573  014424  004037  017374                JSR     R0,WRT.RP
3574  014430  000000                        RPCS1
3575  014432  014436                        CI7
3576  014434  000761                        BR      CI5
3577  014436  032764  010000  000010  CI7:  BIT     #BIT12,RPCS2(R4)  ;DRIVE NON-EXISTENT ?
3578  014444  001034                        BNE     CI8             ;BR IF YES
3579  014446  005702                  1$:   TST     R2              ;ANYTHING IN QUEUE ?
3580  014450  001405                        BEQ     CI7B            ;BR IF NOT
3581  014452  012762  104000  000016        MOV     #BIT15!BIT11,16(R2)     ;SET "PARITY" ERROR INDICATOR
3582  014460  004737  017566                JSR     PC,SVRH11       ;GO SAVE THE RH11/RP04/5/6 REGISTERS
3583  014464  012746  000111          CI7B: MOV     #111,-(SP)      ;DO A "DRIVE CLEAR"
3584  014470  004037  017374                JSR     R0,WRT.RP
3585  014474  000000                        RPCS1
3586  014476  014536                        CI8
3587  014500  004737  020326                JSR     PC,EMPTYQ       ;EMPTY THE QUEUE
3588  014504  105061  012166                CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD IN QUEUE FLAG
3589  014510  105061  012110                CLRB    DRVACT(R1)      ;DRIVE IS IDLE
3590  014514  020137  012232                CMP     R1,DTUW         ;IF THIS DRIVE HAD AN I/O REQUEST
3591  014520  001005                        BNE     1$              ;IN PROGRESS CLEAR ALL OF THE FLAGS
3592  014522  005037  012160                CLR     TRNSWT
3593  014526  012737  177777  012232        MOV     #-1,DTUW
3594  014534  000207                  1$:   RTS     PC
3595  014536  104412                  CI8:  SAVREG                  ;SAVE R0 - R5
3596  014540  032764  010000  000010        BIT     #BIT12,RPCS2(R4)  ;IS 'NED' SET ?
3597  014546  001002                        BNE     1$              ;BR IF YES
3598  014550  005001                        CLR     R1
3599  014552  005003                        CLR     R3
3600  014554  105761  012110          1$:   TSTB    DRVACT(R1)      ;DRIVE ACTIVE?
3601  014560  001443                        BEQ     5$              ;BRANCH IF NO
3602  014562  013702  012160                MOV     TRNSWT,R2       ;GET THE "TRANSFER WAIT" QUEUE
3603  014566  020137  012232                CMP     R1,DTUW         ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
3604  014572  001402                        BEQ     2$              ;BRANCH IF YES
3605  014574  004737  020422                JSR     PC,GETREQ       ;GET THE DPB POINTER
3606  014600  005702                  2$:   TST     R2              ;QUEUE ENTRY FOR DRIVE ?
3607  014602  001415                        BEQ     4$              ;BR IF NOT
3608  014604  032764  010000  000010        BIT     #BIT12,RPCS2(R4)  ;'NED' SET ?
3609  014612  001404                        BEQ     3$              ;BR IF NOT
3610  014614  012762  100002  000016        MOV     #BIT15!BIT01,16(R2)  ;SET 'DRIVE NON-EXISTENT' INDICATOR
3611  014622  000405                        BR      4$              ;CONTINUE
3612  014624  012762  102000  000016  3$:   MOV     #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
3613  014632  004737  017566                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
3614  014636  012763  177777  012212  4$:   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
3615  014644  105061  012110                CLRB    DRVACT(R1)      ;SET "DRIVE ACTIVE" TO IDLE
```

```
3616  014650  020137  012232          CMP     R1,DTUW         ;IS THIS DRIVE SETUP FOR A TRANSFER
3617  014654  001005                  BNE     5$              ;BR IF NOT
3618  014656  012737  177777  012232  MOV     #-1,DTUW        ;RESET THE INDICATOR
3619  014664  005037  012160          CLR     TRNSWT          ;CLEAR THE TRANSFER QUEUE
3620  014670  105061  012166  5$:     CLRB    ULDFLG(R1)      ;CLEAR UNLOAD FLAG
3621  014674  032764  010000  000010  BIT     #BIT12,RPCS2(R4)  ;'NED' SET ?
3622  014702  001021                  BNE     6$              ;BR IF YES
3623  014704  005201                  INC     R1              ;MOVE TO THE NEXT DRIVE
3624  014706  062703  000002          ADD     #2,R3
3625  014712  042701  177770          BIC     #^C7,R1
3626  014716  001316                  BNE     1$              ;BRANCH IF MORE DRIVES
3627  014720  012737  177777  012232  MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
3628  014726  005037  012160          CLR     TRNSWT          ;CLEAR THE 'TRANSFER WAIT' QUEUE
3629  014732  004737  020250          JSR     PC,CLRQUE       ;CLEAR ALL OF THE REQUEST QUEUES
3630  014736  012764  000040  000010  MOV     #BIT05,RPCS2(R4)  ;DO A MASSBUS INIT.
3631  014744  000406                  BR      7$              ;CONTINUE
3632  014746  004737  020326  6$:     JSR     PC,EMPTYQ       ;CLEAR THE DRIVE'S QUEUE
3633  014752  105061  012120          CLRB    DRVSTA(R1)      ;SET DRIVE TO OFFLINE
3634  014756  105061  012130          CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
3635  014762  004737  017704  7$:     JSR     PC,SET.IE       ;SET "IE" WITHOUT "TRE"
3636  014766  104413                  RESREG                  ;RESTORE R0 - R5
3637  014770  000207                  RTS     PC              ;RETURN
3638                                  ;LOOK AHEAD ROUTINE
3639                                  ;
3640                                  ;CALL
3641                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
3642                                  ;       MOV     #DPB,R2         ;POINT TO DPB
3643                                  ;       JSR     R0,LA           ;GO CHECK THE WINDOW
3644                                  ;       RETURN1                 ;ERROR RETURN
3645                                  ;       RETURN2                 ;START A SEARCH
3646                                  ;       RETURN3                 ;START A DATA TRANSFER
3647                                  ;
3648
3649  014772  013704  012246  LA:     MOV     RPADR,R4        ;GET RPCS1'S ADDRESS
3650  014776  010164  000010          MOV     R1,RPCS2(R4)    ;SELECT DRIVE
3651  015002  004037  017214          JSR     R0,RD.RP        ;READ CURRENT CYLINDER
3652  015006  000036                  RPCC
3653  015010  015122                  4$                      ;ERROR RETURN ADDRESS
3654  015012  022662  000012          CMP     (SP)+,12(R2)    ;IS CURRENT CYLINDER=DESIRED
3655                                                          ;CYLINDER?
3656  015016  001037                  BNE     3$              ;EXIT IF NO
3657  015020  105261  012176          INCB    LACNT(R1)       ;INCREMENT THE LOOK AHEAD COUNT
3658  015024  126137  012176  012254  CMPB    LACNT(R1),MXLACT  ;EXCEED MAX?
3659  015032  003026                  BGT     2$              ;BRANCH IF YES
3660  015034  116203  000010          MOVB    10(R2),R3       ;GET DESIRED SECTOR ADDRESS AND
3661  015040  000303                  SWAB    R3              ;MULT. BY 64--ALIGN WITH
3662  015042  006203                  ASR     R3              ;LOOK AHEAD REGISTER
3663  015044  006203                  ASR     R3
3664  015046  012737  000340  177776  MOV     #340,@#PS       ;PRIORITY LEVEL "7"
3665  015054  004037  017214          JSR     R0,RD.RP        ;READ LOOK AHEAD REGISTER
3666  015060  000020                  RPLA
3667  015062  015122                  4$
3668  015064  162603                  SUB     (SP)+,R3        ;CALCULATE THE DELTA
3669  015066  002002                  BGE     1$
3670  015070  062703  002600          ADD     #<22.*64.>,R3   ;MAKE THE DELTA POSITIVE
3671  015074  023703  012256  1$:     CMP     MXDLTA,R3       ;CHECK THE DELTA TO SEE
```

```
3672  015100  002406                          BLT     3$              ;IF IT IS WITHIN THE
3673  015102  023703  012260                  CMP     MNDLTA,R3       ;WINDOW---IF YES, ZERO
3674  015106  002003                          BGE     3$              ;THE LOOK AHEAD COUNT
3675  015110  105061  012176          2$:     CLRB    LACNT(R1)       ;AND TAKE THE I/O EXIT
3676  015114  005720                          TST     (R0)+
3677  015116  005720                  3$:     TST     (R0)+           ;ADJUST THE RETURN ADDRESS
3678  015120  000402                          BR      5$              ;EXIT
3679  015122  004737  014436          4$:     JSR     PC,CI7          ;PROCESS THE ERROR
3680  015126  000200                  5$:     RTS     R0              ;RETURN
3681
3682                                  ;INTERRUPT SERVICE ROUTINE
3683
3684  015130  112737  000001  012164  ISR:    MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
3685  015136  104412                          SAVREG                  ;SAVE R0 - R5
3686  015140  013704  012246                  MOV     RPADR,R4        ;ADDRESS OF RHSCS1
3687  015144  013701  012232                  MOV     DTUW,R1         ;GET "DATA TRANSFER UNDERWAY" INDICATOR
3688  015150  002403                          BLT     1$              ;BRANCH IF NO DATA TRANSFER UNDERWAY
3689  015152  004737  015174                  JSR     PC,TD           ;CALL TRANSFER DONE
3690  015156  000402                          BR      2$              ;EXIT
3691  015160  004737  015334          1$:     JSR     PC,SC           ;CALL SPECIAL CONDITIONS
3692  015164  104413                  2$:     RESREG                  ;RESTORE R0 - R5
3693  015166  105037  012164                  CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
3694  015172  000002                          RTI                     ;RETURN
3695
3696                                  ;TRANSFER DONE ROUTINE
3697
3698  015174  105061  012110          TD:     CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
3699  015200  012737  177777  012232          MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
3700  015206  006301                          ASL     R1
3701  015210  012761  177777  012212          MOV     #-1,TIMER(R1)   ;CANCEL TIMEOUT
3702  015216  006201                          ASR     R1
3703  015220  013702  012160                  MOV     TRNSWT,R2       ;GET "DPB" ADDRESS FROM THE
3704  015224  005037  012160                  CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
3705  015230  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET DONE
3706  015236  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
3707  015242  004037  017214                  JSR     R0,RD.RP        ;TRANSFER ERROR(TRE=1)?
3708  015246  000000                          RPCS1
3709  015250  014436                          CI7
3710  015252  006126                          ROL     (SP)+
3711  015254  100413                          BMI     3$              ;BR IF YES
3712  015256  005737  012206                  TST     SAVEFG          ;SAVE THE RH11/RP04/5/6 REGISTERS?
3713  015262  100002                          BPL     1$              ;BRANCH IF NO
3714  015264  004737  017566                  JSR     PC,SVRH11       ;YES--SAVE THE REGISTERS
3715  015270  004737  013326          1$:     JSR     PC,OPT          ;CALL OPTIMIZER
3716  015274  000417                          BR      SC              ;CHECK OTHER DRIVES
3717  015276  012714  000113          2$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
3718  015302  000414                          BR      SC              ;CHECK FOR OTHER DRIVES
3719  015304  052762  100100  000016  3$:     BIS     #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
3720  015312  004737  020326                  JSR     PC,EMPTYQ       ;EMPTY THE "DRIVE'S WAIT" QUEUE
3721  015316  004737  017566                  JSR     PC,SVRH11       ;SAVE THE RH11/RP04/5/6 REGISTERS
3722  015322  012714  040111                  MOV     #40111,(R4)     ;ISSUE A "DRIVE CLEAR"
3723  015326  012714  000113                  MOV     #113,(R4)       ;ISSUE A RELEASE TO THE DRIVE
3724  015332  000400                          BR      SC              ;CHECK FOR OTHER DRIVES
3725
3726                                  ;SPECIAL CONDITION ROUTINE
3727
```

```
3728  015334  116403  000016          SC:     MOVB    RPAS(R4),R3     ;READ "RPAS"
3729  015340  001014                          BNE     2$              ;BRANCH IF ANY 'ATA' BITS SET
3730  015342  004037  017214                  JSR     R0,RD.RP        ;READ CONTROL AND STATUS REGISTER
3731  015346  000000                          RPCS1
3732  015350  014536                          CI8
3733  015352  106126                          ROLB    (SP)+           ;IS "IE"=1?
3734  015354  100405                          BMI     1$              ;YES, NO DRIVES TO CHECK
3735  015356  004037  020506                  JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
3736  015362  104001                          ERROR   1               ;REPORT AN ILLEGAL INTERRUPT
3737  015364  004737  017704                  JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
3738  015370  000207                  1$:     RTS     PC              ;RETURN
3739  015372  005046                  2$:     CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
3740  015374  110316                          MOVB    R3,(SP)         ;AN "ATA"=1
3741  015376  012703  000001                  MOV     #1,R3
3742  015402  005001                          CLR     R1
3743  015404  030316                  SC3:    BIT     R3,(SP)         ;ATA=1?
3744  015406  001005                          BNE     SC5             ;YES--BRANCH
3745  015410  005201                  SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
3746  015412  106303                          ASLB    R3
3747  015414  001373                          BNE     SC3             ;BRANCH IF MORE TO CHECK?
3748  015416  005726                          TST     (SP)+           ;CLEAN OFF THE STACK
3749  015420  000207                          RTS     PC              ;RETURN TO USER
3750  015422  105761  012140          SC5:    TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
3751  015426  001402                          BEQ     1$              ;BR IF NOT
3752  015430  000137  016326                  JMP     SC13            ;PROCESS THE DRIVE
3753  015434  105761  012150          1$:     TSTB    DPRQS(R1)       ;PORT REQUEST OUTSTANDING ?
3754  015440  001402                          BEQ     2$              ;BR IF NOT
3755  015442  000137  016326                  JMP     SC13            ;START THE OUTSTANDING COMMAND
3756  015446  105761  012120          2$:     TSTB    DRVSTA(R1)      ;CHECK THE DRIVE STATUS
3757  015452  003025                          BGT     5$              ;BRANCH IF ONLINE
3758  015454  105761  012166                  TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS?
3759  015460  003422                          BLE     5$              ;BRANCH IF NOT
3760  015462  004737  020422                  JSR     PC,GETREQ       ;GET DPB POINTER
3761  015466  004737  017566                  JSR     PC,SVR411       ;SAVE THE RH11/RP04/5/6 REGISTERS
3762  015472  004737  016256                  JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
3763                                                                  ;ALSO DO A DRIVE INIT (DRVINT)
3764  015476  105761  012120                  TSTB    DRVSTA(R1)      ;DID DRIVE COME ONLINE?
3765  015502  003416                          BLE     6$              ;NO---BRANCH
3766  015504  032737  040000  012100          BIT     #BIT14,RPERRS   ;WAS THERE AN ERROR?
3767  015512  001002                          BNE     3$              ;BR IF ERROR
3768  015514  000137  016166                  JMP     SC11            ;NO ERROR
3769  015520  013705  012102          3$:     MOV     RPERRS+2,R5     ;YES -- PICKUP RPER1 AND
3770  015524  000502                          BR      SCFA            ;GO PROCESS THE ERROR
3771  015526  105761  012110          5$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
3772  015532  001033                          BNE     SC6             ;BR IF EITHER
3773  015534  004737  016256                  JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
3774                                                                  ;ALSO DO A DRVINT
3775  015540  105761  012140          6$:     TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE ?
3776  015544  001321                          BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
3777  015546  105761  012120                  TSTB    DRVSTA(R1)      ;CHECK ON DRIVE'S STATUS
3778  015552  100412                          BMI     7$              ;BR IF UNSAFE
3779  015554  032737  020000  012106          BIT     #BIT13,RPERRS+6 ;ADDRESS PLUG CHANGED ?
3780  015562  001013                          BNE     8$              ;BR IF YES
3781  015564  012746  000113                  MOV     #113,-(SP)      ;RELEASE COMMAND
3782  015570  004037  017374                  JSR     R0,WRT.RP       ;WRITE THE COMMAND INTO RPCS1
3783  015574  000000                          RPCS1                   ;REGISTER INDEX
```

```
3784  015576  016136                        SC8                     ;PARITY EXIT ADDRESS
3785  015600  011605              7$:       MOV     (SP),R5         ;PICKUP (RPAS) BEFORE THE ERROR CALL
3786  015602  004037  020506                JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
3787  015606  104002                        ERROR   2               ;REPORT THE UNEXPECTED ATTENTION
3788  015610  000677                        BR      SC4             ;GO CHECK FOR MORE ATA'S
3789  015612                        8$:
3790  015612  004037  020506                JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
3791  015616  104005                        ERROR   5               ;REPORT THE ADDRESS PLUG CHANGE
3792  015620  000673                        BR      SC4             ;CHECK FOR MORE DRIVES
3793  015622  006301              SC6:      ASL     R1              ;SETUP TO ADDRESS WORDS
3794  015624  012761  177777  012212        MOV     #-1,TIMER(R1)   ;STOP THE TIMER
3795  015632  006201                        ASR     R1              ;RESTORE THE DRIVE ADDRESS
3796  015634  004737  020422                JSR     PC,GETREQ       ;GET THE DPB POINTER FROM THE QUEUE
3797  015640  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
3798  015644  004037  017214                JSR     R0,RD.RP        ;READ THE RP04'S STATUS REG.
3799  015650  000012                        RPDS1
3800  015652  016136                        SC8
3801  015654  011605                        MOV     (SP),R5         ;AND PUT IT IN R5
3802  015656  006126                        ROL     (SP)+           ;WAS THERE AN ERROR?
3803  015660  100407                        BMI     1$              ;BR IF ERROR
3804  015662  105761  012110                TSTB    DRVACT(R1)      ;CHECK DRIVE'S STATE
3805  015666  003137                        BGT     SC11            ;BR IF DRIVE ACTIVE WITH ORDER
3806  015670  052762  100210  000016        BIS     #BIT15!BIT07!BIT03,16(R2)  ;INFORM USER OF ERROR RECOVER COMPLETION
3807  015676  000470                        BR      SC7
3808  015700  004037  017214        1$:     JSR     R0,RD.RP        ;READ ERROR REGISTER #1
3809  015704  000014                        RPER1
3810  015706  016136                        SC8
3811  015710  012605                        MOV     (SP)+,R5        ;AND SAVE IT IN R5
3812  015712  004737  017566                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
3813  015716  012746  000111                MOV     #111,-(SP)      ;ISSUE A DRIVE CLEAR
3814  015722  004037  017374                JSR     R0,WRT.RP
3815  015726  000000                        RPCS1
3816  015730  016136                        SC8
3817  015732  006105              SC6A:     ROL     R5              ;WAS "UNSAFE" CONDITION =1?
3818  015734  100406                        BMI     1$              ;BRANCH IF YES
3819  015736  005702                        TST     R2              ;ANYTHING IN QUEUE ?
3820  015740  001447                        BEQ     SC7             ;BR IF NOT
3821  015742  052762  100240  000016        BIS     #BIT15!BIT07!BIT05,16(R2)       ;INFORM USER OF ERROR
3822  015750  000443                        BR      SC7
3823  015752  004037  017214        1$:     JSR     R0,RD.RP        ;READ DRIVE STATUS REG. #1
3824  015756  000012                        RPDS1
3825  015760  016136                        SC8
3826  015762  011605                        MOV     (SP),R5         ;SAVE RPDS1 IN R5
3827  015764  006126                        ROL     (SP)+           ;"ERR"=1?
3828  015766  100011                        BPL     2$              ;BR IF NO--UNSAFE CLEARED
3829  015770  112761  177777  012120        MOVB    #-1,DRVSTA(R1)  ;DRIVE IS UNSAFE
3830  015776  004737  017566                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
3831  016002  052762  110000  000016        BIS     #BIT15!BIT12,16(R2)     ;INFORM USER OF UNSAFE ERROR
3832  016010  000423                        BR      SC7
3833  016012  032705  010000        2$:     BIT     #BIT12,R5       ;"MOL" = 1 ?
3834  016016  001015                        BNE     3$              ;BR IF YES
3835  016020  112761  177777  012110        MOVB    #-1,DRVACT(R1)  ;ACTIVE ERROR RECOVER
3836  016026  112761  000001  012120        MOVB    #1,DRVSTA(R1)   ;ONLINE
3837  016034  006301                        ASL     R1
3838  016036  012761  072460  012212        MOV     #30000.,TIMER(R1) ;START 30 SECOND TIMER
3839  016044  006201                        ASR     R1
```

```
3840  016046  000137  015410                    JMP     SC4
3841  016052  052762  100220  000016  3$:       BIS     #BIT15!BIT07!BIT04,16(R2)       ;INFORM USER OF ERROR
3842  016060  105061  012110          SC7:      CLRB    DRVACT(R1)              ;DRIVE IS IDLE
3843  016064  004737  020326                    JSR     PC,EMPTYQ               ;DUMP THE QUEUE
3844  016070  105761  012166                    TSTB    ULDFLG(R1)              ;UNLOAD IN PROGRESS OR QUEUE?
3845  016074  003002                            BGT     1$                      ;BR IF NOT
3846  016076  105061  012166                    CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
3847  016102  116164  012234  000016  1$:       MOVB    ATABIT(R1),RPAS(R4)   ;CLEAR ATTENTION BIT
3848  016110  105761  012120                    TSTB    DRVSTA(R1)              ;IS THE DRIVE UNSAFE ?
3849  016114  100406                            BMI     2$                      ;BR IF IT IS
3850  016116  012746  000113                    MOV     #113,-(SP)              ;RELEASE COMMAND
3851  016122  004037  017374                    JSR     R0,WRT.RP               ;WRITE THE COMMAND INTO RPCS1
3852  016126  000000                            RPCS1                           ;REGISTER INDEX
3853  016130  016136                            SC8                             ;PARITY EXIT ADDRESS
3854  016132  000137  015410          2$:       JMP     SC4                     ;CHECK FOR MORE DRIVES
3855  016136  105761  012110          SC8:      TSTB    DRVACT(R1)              ;IS DRIVE IDLE?
3856  016142  001405                            BEQ     1$                      ;YES--BRANCH
3857  016144  004737  020422                    JSR     PC,GETREQ               ;GET DPB POINTER
3858  016150  004737  014436                    JSR     PC,CI7                  ;PROCESS THE PARITY ERROR
3859  016154  000402                            BR      2$                      ;CONTINUE
3860  016156  004737  014464          1$:       JSR     PC,CI7B                 ;PROCESS THE UNCORRECTABLE PARITY ERROR
3861  016162  000137  015410          2$:       JMP     SC4                     ;CHECK MORE DRIVES
3862  016166  105761  012166          SC11:     TSTB    ULDFLG(R1)              ;"UNLOAD IN PROGRESS"?
3863  016172  003402                            BLE     1$                      ;BRANCH IF NO
3864  016174  105061  012166                    CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
3865  016200  105061  012110          1$:       CLRB    DRVACT(R1)              ;SET DRIVE IDLE
3866  016204  136137  012234  012162            BITB    ATABIT(R1),SRCHWT     ;DOING A SEARCH OPERATION FOR
3867                                                                            ;AN I/O COMMAND?
3868  016212  001012                            BNE     2$                      ;BRANCH IF YES
3869  016214  004737  020444                    JSR     PC,POPQUE               ;REMOVE REQUEST FROM QUEUE
3870  016220  052762  000200  000016            BIS     #BIT07,16(R2)           ;SET "DONE" BIT
3871  016226  005737  012206                    TST     SAVEFG                  ;SAVE THE REGISTERS?
3872  016232  100002                            BPL     2$                      ;BRANCH IF NO
3873  016234  004737  017566                    JSR     PC,SVRH11               ;YES--SAVE ALL OF THE RH11/RP04/5/6 REG'S
3874  016240  116164  012234  000016  2$:       MOVB    ATABIT(R1),RPAS(R4)   ;CLEAR ATTENTION BIT
3875  016246  004737  013326                    JSR     PC,OPT                  ;START A REQUEST
3876  016252  000137  015410                    JMP     SC4                     ;CHECK FOR MORE DRIVES
3877  016256  010164  000010          SC12:     MOV     R1,RPCS2(R4)            ;SELECT DRIVE
3878  016262  016437  000012  012100            MOV     RPDS1(R4),RPERRS        ;SAVE THE FOUR REGISTERS THAT
3879  016270  016437  000014  012102            MOV     RPER1(R4),RPERRS+2      ;WILL TELL US SOMETHING
3880  016276  016437  000040  012104            MOV     RPER2(R4),RPERRS+4
3881  016304  016437  000042  012106            MOV     RPER3(R4),RPERRS+6
3882  016312  004037  012476                    JSR     R0,DRVINT               ;INIT. THE STATE OF THE DRIVE
3883  016316  000401                            BR      1$                      ;TAKE ERROR EXIT
3884  016320  000207                            RTS     PC                      ;RETURN
3885  016322  005726                  1$:       TST     (SP)+                   ;POP PC OFF OF THE STACK
3886  016324  000704                            BR      SC8                     ;PROCESS THE PARITY ERROR
3887  016326  006301                  SC13:     ASL     R1                      ;SETUP TO ADDRESS WORDS
3888  016330  012761  177777  012212            MOV     #-1,TIMER(R1)           ;STOP THE TIMER
3889  016336  006201                            ASR     R1                      ;
3890  016340  010164  000010                    MOV     R1,RPCS2(R4)            ;SELECT THE DRIVE
3891  016344  116164  012234  000016            MOVB    ATABIT(R1),RPAS(R4)   ;CLEAR THE ATTENTION BIT
3892  016352  032714  004000                    BIT     #BIT11,(R4)             ;DRIVE AVAILABLE ?
3893  016356  001006                            BNE     1$                      ;BR IF AVAILABLE
3894  016360  006301                            ASL     R1
3895
```

```
3896  016362  012761  023420  012212          MOV     #10000.,TIMER(R1)  ;START 10 SEC TIMER AGAIN
3897
3898  016370  006201                          ASR     R1
3899  016372  000433                          BR      3$                 ;EXIT
3900  016374  105761  012140          1$:     TSTB    DPINT(R1)          ;INITIALIZING THE DRIVE ?
3901  016400  001424                          BEQ     2$                 ;BR IF NOT
3902  016402  105061  012140                  CLRB    DPINT(R1)          ;CLEAR THE INIT INDICATOR
3903  016406  004037  012476                  JSR     R0,DRVINT          ;GO INIT THE DRIVE
3904  016412  000240                          NOP                        ;DUMMY PARITY ERROR RETURN
3905  016414  105761  012120                  TSTB    DRVSTA(R1)         ;DRIVE ONLINE ?
3906  016420  003014                          BGT     2$                 ;BR IF YES -- START ORDER
3907  016422  005702                          TST     R2                 ;QUEUE ENTRY FOR THE DRIVE
3908  016424  001416                          BEQ     3$                 ;BR IF NOT
3909  016426  004737  020422                  JSR     PC,GETREQ          ;GET DPB ADDRESS
3910  016432  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)  ;INFORM USER THAT DRIVE OFFLINE
3911  016440  004737  017566                  JSR     PC,SVRH11          ;SAVE THE REGISTERS
3912  016444  004737  020326                  JSR     PC,EMPTYQ          ;EMPTY THE REQUEST QUEUE
3913  016450  000404                          BR      3$
3914  016452  105061  012150          2$:     CLRB    DPRQS(R1)          ;CLEAR THE PORT REQUEST INDICATOR
3915  016456  004737  013326                  JSR     PC,OPT             ;START THE PENDING REQUEST
3916  016462  000137  015410          3$:     JMP     SC4                ;PROCESS OTHER DRIVES
3917
3918                                  ;RP04/5/6 TIMER ROUTINE
3919                                  ;CALL
3920                                  ;       MOV     #TIME,-(SP)        ;ELASPED TIME IN MILLISECONDS ON THE STACK
3921                                  ;       JSR     PC,RPTMR           ;CALL RP04/5/6 TIME ROUTINE
3922
3923  016466  005737  012164          RPTMR:  TST     ACTDRV             ;CHECK "ACTDRV & ACTSTR"
3924  016472  001030                          BNE     4$                 ;IF NON ZERO EXIT
3925  016474  112737  000001  012165          MOVB    #1,ACTSTR          ;SET "ACTSTR"
3926  016502  104412                          SAVREG                     ;SAVE R0 - R5
3927  016504  005001                          CLR     R1                 ;START WITH DRIVE 0
3928  016506  005003                          CLR     R3
3929  016510  005763  012212          1$:     TST     TIMER(R3)          ;IS THE TIMER RUNNING?
3930  016514  002407                          BLT     2$                 ;BRANCH IF NO
3931  016516  166663  000002  012212          SUB     2(SP),TIMER(R3)    ;COUNT THE INTERVAL
3932  016524  003003                          BGT     2$                 ;BR IF NO SOFTWARE TIMEOUT
3933  016526  004737  016560                  JSR     PC,STO             ;CALL SOFTWARE TIMEOUT ROUTINE
3934  016532  000405                          BR      3$                 ;GO TO THE EXIT
3935  016534  005201                  2$:     INC     R1                 ;MOVE TO NEXT DRIVE
3936  016536  005723                          TST     (R3)+
3937  016540  022701  000010                  CMP     #8.,R1             ;OUT OF DRIVES?
3938  016544  003361                          BGT     1$                 ;BRANCH IF NO
3939  016546  104413                  3$:     RESREG                     ;RESTORE R0 - R5
3940  016550  105037  012165                  CLRB    ACTSTR             ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
3941  016554  012616                  4$:     MOV     (SP)+,(SP)         ;ADJUST THE STACK
3942  016556  000207                          RTS     PC                 ;RETURN
3943
3944                                  ;SOFTWARE TIMEOUT ROUTINE
3945                                  ;
3946                                  ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
3947                                  ;       OR GREATER
3948                                  ;
3949                                  ;CALL:  STO
3950                                  ;       MOV     #DRVNUM,R1         ;DRIVE NUMBER
3951                                  ;       JSR     PC,STO             ;CALL
```

```
                                        ;       RETURN
                                        ;
                                        STO:    MOV     R1,-(SP)        ;SAVE R1
                                                MOV     R3,-(SP)        ;SAVE R3
                                                MOV     RPADR,R4        ;GET ADDRESS OF "RPCS1"
                                                MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
                                                JSR     R0,RD.RP        ;READ "DRIVE STATUS REG"
                                                RPDS1
                                                STO5
                                                TSTB    (SP)+           ;IS "DRY"=1?
                                                BMI     STO2            ;BR IF YES
                                        STO1:   TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
                                                BNE     STO2            ;BR IF YES
                                                TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
                                                BNE     STO2            ;BR IF YES
                                                MOV     TRNSWT,R2       ;PICKUP TRANSFER WAIT QUEUE
                                                CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
                                                BEQ     1$              ;BRANCH IF YES
                                                JSR     PC,GETREQ       ;GET DPB ADDRESS
                                        1$:     BIS     #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
                                                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
                                                MOV     #BIT05,RPCS2(R4) ;"INIT" THE MASS BUS
                                                CLRB    DRVACT(R1)      ;DRIVE IS IDLE
                                                CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
                                                CLR     R1              ;START WITH DRIVE 0
                                                CLR     R3
                                        2$:     JSR     R0,DRVINT       ;INIT. THIS DRIVE
                                                BR      STO5            ;PARITY ERROR RETURN
                                                TSTB    DRVACT(R1)      ;DRIVE IDLE BEFORE THE INIT.?
                                                BEQ     4$              ;YES--BRANCH
                                                MOV     TRNSWT,R2       ;GET TRANSFER WAIT QUEUE
                                                CMP     DTUW,R1         ;WAS THERE I/O ON THIS DRIVE?
                                                BEQ     3$              ;YES--BRANCH
                                                JSR     PC,GETREQ       ;GET THE DPB POINTER FROM QUEUE
                                        3$:     BIS     #BIT15!BIT08,16(R2) ;INFORM USER OF INIT.
                                                CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE TO IDLE
                                        4$:     CLRB    ULDFLG(R1)      ;NO UNLOAD
                                                MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                                TST     (R3)+           ;UPDATE THE INDEX
                                                INC     R1              ;INCREMENT THE DRIVE NUMBER
                                                CMP     #8.,R1          ;LAST DRIVE BEEN CHECKED?
                                                BGT     2$              ;NO--LOOP
                                                MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
                                                CLR     TRNSWT          ;CLEAR TRANSFER WAIT QUEUE
                                                JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
                                                BR      STO9            ;EXIT
                                        STO2:   MOVB    RPAS(R4),R5     ;READ ATTENTION REG
                                                BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
                                                BNE     STO3            ;YES--BRANCH
                                                TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
                                                BNE     STO6            ;BR IF YES - DRIVE NOT ONLINE
                                                TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
                                                BNE     STO7            ;BR IF YES - NO RESPONSE TO REQUEST
                                                CMP     R1,DTUW         ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
                                                BNE     STO1            ;BR IF NO
                                                JSR     R0,RD.RP        ;YES--CHECK "RDY"
```

```
                                        RPCS1
                                        STO5
                                        TSTB    (SP)+
                                        BPL     ST01            ;BR IF "RDY"=0
                        ST03:           TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
                                        BNE     1$              ;BR IF INIT PENDING
                                        TSTB    DPRQS(R1)       ;PORT REQUEST PENDING ?
                                        BEQ     ST09            ;BR IF NOT
012212          1$:                     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
                                        BR      ST09            ;EXIT
014536          ST05:                   JSR     PC,CI8          ;GO HANDLE THE PARITY ERROR
                                        BR      ST09
                        ST06:           CLRB    DPINT(R1)       ;CLEAR THE INITIALIZE INDICATOR
                                        CLRB    DRVSTA(R1)      ;SET UNIT OFFLINE
012212                                  MOV     #-1,TIMER(R3)   ;STOP THE TIMER
020422                                  JSR     PC,GETREQ       ;GET THE DPB ADDRESS
                                        TST     R2              ;REQUEST IN QUEUE ?
                                        BEQ     ST09            ;BR IF NOT
000016                                  BIS     #BIT15!BIT14,16(R2)  ;INFORM THE USER DRIVE NOT AVAILABLE
                                        BR      ST08            ;FINISH
012212          ST07:                   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
012150                                  CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST INDICATOR
020422                                  JSR     PC,GETREQ       ;GET DPB ADDRESS
                                        TST     R2              ;QUEUE ENTRY FOR DRIVE ?
                                        BEQ     ST09            ;BR IF NONE
000016                                  MOV     #BIT15!BIT2,16(R2)  ;INFORM USER OF PORT REQUEST ERROR
020326          ST08:                   JSR     PC,EMPTYQ       ;CLEAR THE QUEUE FOR THE DRIVE
017566                                  JSR     PC,SVRH11       ;SAVE THE REGISTERS
                        ST09:           MOV     (SP)+,R3        ;RESTORE R3
                                        MOV     (SP)+,R1        ;RESTORE R1
                                        RTS     PC              ;RETURN

                        ;ROUTINE TO READ A RH11/RP04/5/6 REGISTER
                        ;
                        ;CALL
                                JSR     R0,RD.RP        ;GO READ A REGISTER
                                INDEX                   ;REG. INDEX FROM BASE
                                ERRADR                  ;ERROR ADDRESS--PROCESS ERROR STARTING
                                                        ;AT THIS ADDRESS
                                RETURN                  ;CONTENTS OF REG. IS ON THE STACK
                        ;
012244  017362  RD.RP:          MOV     MCPEMX,RD.RP2   ;MAX. RETRYS ALLOWED
                                MOV     (SP),-(SP)      ;SAVE R0 FOR RETURN
012246  017240                 MOV     RPADR,RD.ADR    ;FORM THE DESIRED ADDRESS
017240                          ADD     (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
                        RD.RP1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RP04
000000  RD.ADR: .WORD   0               ;ADDRESS IS FORMED HERE
000000  RD.WRD: .WORD   0               ;REG. CONTENTS PUT HERE
017242  000002                  MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
012246                          MOV     RPADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
000010                          ADD     #RPCS2,(SP)     ;FORM THE ADDRESS OF RPCS2
010000                          BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
                                BNE     RD.RP3          ;BR IF DRIVE NON-EXISTENT
172752                          MOV     @RPADR,-(SP)    ;READ RPCS1
020000                          BIT     #BIT13,(SP)     ;DID MCPE SET?
                                BNE     1$              ;BRANCH IF YES
```

```
4064   017302   022620                          CMP     (SP)+,(R0)+     ;ADJUST FOR RETURN
4065   017304   000432                          BR      RD.RP4          ;EXIT
4066   017306                           1$:
4067   017306   004037   020506                 JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
4068   017312   104003                          ERROR   3               ;REPORT "MCPE" ERROR
4069   017314   005737   012232                 TST     DTUW            ;DATA TRANSFER UNDERWAY?
4070   017320   100405                          BMI     2$              ;NO--BRANCH
4071   017322   032716   040000                 BIT     #BIT14,(SP)     ;NO--"TRE"=1?
4072   017326   001402                          BEQ     2$              ;NO--BRANCH
4073   017330   005726                          TST     (SP)+           ;YES--CLEAN OFF THE STACK AND
4074   017332   000415                          BR      RD.RP3          ;TAKE THE FATAL ERROR EXIT
4075   017334   052716   040000         2$:     BIS     #BIT14,(SP)     ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
4076   017340   000316                          SWAB    (SP)            ;POSITION BEFORE WRITING
4077   017342   013737   012246   017356        MOV     RPADR,3$        ;FORM ADDRESS OF HIGH BYTE
4078   017350   005237   017356                 INC     3$
4079   017354   112637                          MOVB    (SP)+,@(PC)+    ;WRITE THE HIGH BYTE OF RPCS1
4080   017356   000000                  3$:     .WORD   0               ;ADDRESS STORAGE
4081   017360   005327                          DEC     (PC)+           ;EXCEEDED MAX. RETRYS
4082   017362   000003                  RD.RP2: .WORD   3
4083   017364   002324                          BGE     RD.RP1          ;BRANCH IF NO
4084   017366   011000                  RD.RP3: MOV     (R0),R0         ;FATAL ERROR EXIT
4085   017370   012616                          MOV     (SP)+,(SP)
4086   017372   000200                  RD.RP4: RTS     R0
4087
4088                                    ;ROUTINE TO WRITE A REGISTER
4089                                    ;
4090                                    ;CALL
4091                                    ;       MOV     DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
4092                                    ;       JSR     R0,WRT.RP       ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
4093                                    ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
4094                                    ;       ERRADR                  ;ADDRESS TO RETURN TO ON AN ERROR
4095                                    ;       RETURN                  ;ERROR FREE RETURN
4096
4097   017374   013737   012244   017550  WRT.RP: MOV     MCPEMX,WRT.R2   ;MAX RETRYS ALLOWED
4098   017402   016637   000002   017462          MOV     2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
4099   017410   012616                           MOV     (SP)+,(SP)      ;ADJUST THE STACK
4100   017412   012037   017464                  MOV     (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
4101   017416   001015                           BNE     1$              ;BRANCH IF NOT RPCS1
4102   017420   122737   000150   017462         CMPB    #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
4103   017426   002411                           BLT     1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
4104   017430   004037   017214                  JSR     R0,RD.RP        ;NO---COMBINE A16&A17, & PSEL WITH
4105   017434   000000                           RPCS1                   ;THE COMMAND BEFORE SENDING IT TO
4106   017436   017556                           WRT.R3                  ;THE RH11/RP04
4107   017440   000316                           SWAB    (SP)
4108   017442   042716   177770                  BIC     #^C7,(SP)
4109   017446   112637   017463                  MOVB    (SP)+,WRT.WD+1
4110   017452   063737   012246   017464  1$:     ADD     RPADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
4111   017460   012737                   WRT.R1: MOV     (PC)+,@(PC)+    ;LOAD THE DESIRED REG.
4112   017462   000000                   WRT.WD: .WORD   0               ;WORD TO WRITE GOES HERE
4113   017464   000000                   WRT.AD: .WORD   0               ;ADDRESS IS FORMED HERE
4114   017466   013746   012246                  MOV     RPADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
4115   017472   062716   000010                  ADD     #RPCS2,(SP)     ;FORM THE ADDRESS OF RPCS2
4116   017476   032736   010000                  BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
4117   017502   001025                           BNE     WRT.R3          ;BR IF DRIVE NON-EXISTENT
4118   017504   004037   017214                  JSR     R0,RD.RP        ;CHECK FOR PARITY ERROR ON WRITE
4119   017510   000014                           RPER1
```

```
                                        WRT.R3
                                        BIT     #BIT03,(SP)+
                                        BEQ     WRT.R4          ;BRANCH IF "PAR=0"
                                        MOV     -2(R0),1$       ;PICKUP THE INDEX
                                        JSR     R0,RD.RP        ;READ THE REG.
                                1$:     .WORD   0               ;REG. INDEX
                                        WRT.R3                  ;RETURN TO THIS ADDRESS ON ERROR
                                        JSR     R0,ES.SAV       ;SAVE THE ADDRESS IN '$ESCAPE'
                                        ERROR   4               ;REPORT THE PARITY ON WRITE ERROR
                                        DEC     (PC)+           ;DECREMENT THE ERROR COUNT
                                WRT.R2: .WORD   3               ;RETRY COUNTER
                                        BGE     WRT.R1          ;TRY AGAIN IF NOT FINISHED
                                        TST     (SP)+           ;CLEAN OFF THE STACK
                                WRT.R3: MOV     (R0),R0         ;TAKE THE "PARITY ON WRITE" ERROR EXIT
                                        BR      WRT.R5          ;EXIT
                                WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
                                WRT.R5: RTS     R0

                                ;ROUTINE TO SAVE THE RH11/RP04/5/6 REGISTERS AS PER DPB+14
                                ;
                                ;CALL
                                ;       MOV     #DPBNUM,R2      ;DPB POINTER TO R2
                                ;       JSR     PC,SVRH11       ;SAVE THE DRIVES REG'S

                                SVRH11: SAVREG                  ;SAVE R0 - R5
                                        TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
                                        BEQ     4$              ;BR IF NONE
                                        MOV     RPADR,R4
                                        MOVB    (R2),RPCS2(R4)  ;SELECT DRIVE
                                        MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
                                        BEQ     6$              ;EXIT IF NO ADDRESS
                                        CLR     3$              ;COUNTER & POINTER
                                1$:     CMP     3$,#RPDB        ;REACHED THE BUFFER REGISTER ?
                                        BNE     2$              ;BR IF NOT
                                        BIT     #BIT07,RPCS2(R4)  ;'OR' SET ?
                                        BNE     2$              ;BR IF SET
                                        CLR     (R3)+           ;STORE RPDB AS ZEROES
                                        BR      4$              ;CONTINUE
                                2$:     JSR     R0,RD.RP        ;READ THE SELECTED REGISTER
                                3$:     .WORD   0               ;REGISTER INDEX
                                        5$                      ;ERROR RETURN ADDRESS
                                        MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
                                4$:     CMP     3$,#RPEC2       ;REACHED THE END ?
                                        BEQ     6$              ;BR IF YES
                                        ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
                                        BR      1$              ;CONTINUE READING THE REGISTERS
                                5$:     JSR     PC,CI7          ;PROCESS THE UNCORRECTABLE PARITY ERROR
                                6$:     RESREG                  ;RESTORE R0 - R5
                                        RTS     PC              ;RETURN

                                ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
                                ;CALL
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                ;       JSR     PC,SET.IE       ;SET "IE"
                                ;       RETURN
```

```
4176  017704  010446                SET.IE: MOV     R4,-(SP)        ;SAVE R4
4177  017706  013704  012246                MOV     RPADR,R4        ;PICKUP ADDRESS OF RPCS1
4178  017712  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
4179  017716  011446                        MOV     (R4),-(SP)      ;READ RPCS1
4180  017720  052716  040000                BIS     #BIT14,(SP)     ;SET THE "TRE" BIT OF THE WORD READ
4181  017724  000316                        SWAB    (SP)            ;ADJUST FOR DATO
4182  017726  112714  000100                MOVB    #BIT06,(R4)     ;SET "IE"
4183  017732  032764  010000  000010        BIT     #BIT12,RPCS2(R4) ;IS "NED"=1?
4184  017740  001002                        BNE     1$              ;YES--CLEAR "TRE"
4185  017742  005726                        TST     (SP)+           ;CLEAN OFF THE STACK
4186  017744  000402                        BR      2$
4187  017746  112664  000001        1$:     MOVB    (SP)+,1(R4)     ;CLEAR "TRE"
4188  017752  012604                2$:     MOV     (SP)+,R4        ;RESTORE R4
4189  017754  000207                        RTS     PC              ;RETURN TO CALLER
4190
4191                                ;QUEUE COUNT
4192  017756     000                QCNT:   .BYTE   0               ;DRIVE 0
4193  017757     000                        .BYTE   0               ;DRIVE 1
4194  017760     000                        .BYTE   0               ;DRIVE 2
4195  017761     000                        .BYTE   0               ;DRIVE 3
4196  017762     000                        .BYTE   0               ;DRIVE 4
4197  017763     000                        .BYTE   0               ;DRIVE 5
4198  017764     000                        .BYTE   0               ;DRIVE 6
4199  017765     000                        .BYTE   0               ;DRIVE 7
4200
4201
4202                                ;QUEUE INPUT POINTERS
4203  017766  020050                QINPT:  .WORD   QDRV0           ;DRIVE 0
4204  017770  020070                        .WORD   QDRV1           ;DRIVE 1
4205  017772  020110                        .WORD   QDRV2           ;DRIVE 2
4206  017774  020130                        .WORD   QDRV3           ;DRIVE 3
4207  017776  020150                        .WORD   QDRV4           ;DRIVE 4
4208  020000  020170                        .WORD   QDRV5           ;DRIVE 5
4209  020002  020210                        .WORD   QDRV6           ;DRIVE 6
4210  020004  020230                        .WORD   QDRV7           ;DRIVE 7
4211
4212
4213                                ;QUEUE OUTPUT POINTERS
4214  020006  020050                QOUTPT: .WORD   QDRV0           ;DRIVE 0
4215  020010  020070                        .WORD   QDRV1           ;DRIVE 1
4216  020012  020110                        .WORD   QDRV2           ;DRIVE 2
4217  020014  020130                        .WORD   QDRV3           ;DRIVE 3
4218  020016  020150                        .WORD   QDRV4           ;DRIVE 4
4219  020020  020170                        .WORD   QDRV5           ;DRIVE 5
4220  020022  020210                        .WORD   QDRV6           ;DRIVE 6
4221  020024  020230                        .WORD   QDRV7           ;DRIVE 7
4222
4223  020026  020050                QSTART: .WORD   QDRV0           ;DRIVE 0 START ADDRESS
4224  020030  020070                QSTOP:  .WORD   QDRV1           ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
4225  020032  020110                        .WORD   QDRV2           ;STOP DRIVE 1--START DRIVE 2
4226  020034  020130                        .WORD   QDRV3           ;STOP DRIVE 2--START DRIVE 3
4227  020036  020150                        .WORD   QDRV4           ;STOP DRIVE 3--START DRIVE 4
4228  020040  020170                        .WORD   QDRV5           ;STOP DRIVE 4--START DRIVE 5
4229  020042  020210                        .WORD   QDRV6           ;STOP DRIVE 5--START DRIVE 6
4230  020044  020230                        .WORD   QDRV7           ;STOP DRIVE 6--START DRIVE 7
4231  020046  020250                        .WORD   QTERM           ;STOP DRIVE 7
```

```
                                ;DRIVE REQUEST QUEUES
                                ;
020050  000010                  QDRV0:  .BLKW   10
020070  000010                  QDRV1:  .BLKW   10
020110  000010                  QDRV2:  .BLKW   10
020130  000010                  QDRV3:  .BLKW   10
020150  000010                  QDRV4:  .BLKW   10
020170  000010                  QDRV5:  .BLKW   10
020210  000010                  QDRV6:  .BLKW   10
020230  000010                  QDRV7:  .BLKW   10
        020250                  QTERM=.
                                ;
                                ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
                                ;
                                ;CALL
                                ;       JSR     PC,CLRQUE
                                ;
020250  104412                  CLRQUE: SAVREG                  ;SAVE R0 - R5
020252  012702  017756                  MOV     #QCNT,R2        ;ZERO THE QUEUE COUNTS
020256  005022                          CLR     (R2)+           ;DRIVES 0 & 1
020260  005022                          CLR     (R2)+           ;DRIVES 2 & 3
020262  005022                          CLR     (R2)+           ;DRIVES 4 & 5
020264  005022                          CLR     (R2)+           ;DRIVES 6 & 7
020266  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING
020272  012701  020026                  MOV     #QSTART,R1      ;ADDRESS OF THE QUEUE INTO
020276  012122                  1$:     MOV     (R1)+,(R2)+     ;THE QUEUE INPUT POINTER
020300  005303                          DEC     R3
020302  001375                          BNE     1$
020304  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING ADDRESS
020310  012701  020026                  MOV     #QSTART,R1      ;OF THE QUEUE INTO THE
020314  012122                  2$:     MOV     (R1)+,(R2)+     ;QUEUE OUTPUT POINTER
020316  005303                          DEC     R3
020320  001375                          BNE     2$
020322  104413                          RESREG                  ;RESTORE R0 - R5
020324  000207                          RTS     PC
                                ;
                                ;EMPTY THE QUEUE SPECIFIED BY R1
                                ;
                                ;CALL
                                ;       MOV     DRVNUM,R1       ;DRIVE NUMBER TO R1
                                ;       JSR     PC,EMPTYQ
                                ;
020326  105061  017756          EMPTYQ: CLRB    QCNT(R1)        ;CLEAR NUMBER OF ITEMS IN QUEUE
020332  006301                          ASL     R1
020334  016161  017766  020006          MOV     QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
020342  006201                          ASR     R1
020344  000207                          RTS     PC
                                ;
                                ;ROUTINE TO PUT A REQUEST IN QUEUE
                                ;
                                ;CALL
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                ;       MOV     #DPB,R2         ;ADDRESS OF PARAMETER BLOCK
                                ;       JSR     R0,DRVQUE       ;GO PUT REQUEST IN QUEUE
                                ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
```

```
                                        ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE
                                        ;
020346  122761  000010  017756  DRVQUE: CMPB    #10,QCNT(R1)            ;IS QUEUE FULL?
020354  001421                          BEQ     2$                      ;BR IF YES-TAKE RETURN1
020356  105261  017756                  INCB    QCNT(R1)                ;INCREMENT QUEUE COUNT
020362  006301                          ASL     R1
020364  010271  017766                  MOV     R2,@QINPT(R1)           ;PUT THIS REQUEST IN QUEUE
020370  062761  000002  017766          ADD     #2,QINPT(R1)            ;UPDATE THE QUEUE POINTER
020376  026161  017766  020030          CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
020404  001003                          BNE     1$                      ;BRANCH IF NO
020406  016161  020026  017766          MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
020414  006201                  1$:     ASR     R1
020416  005720                          TST     (R0)+                   ;TAKE RETURN 2
020420  000200                  2$:     RTS     R0                      ;RETURN TO USER
                                        ;
                                        ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
                                        ;
                                        ;CALL
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                        ;       JSR     PC,GETREQ       ;GO GET THE REQUEST
                                        ;       RETURN                  ;R2="DPB" ADDRESS OF THE REQUEST
                                        ;                               ;R2=0 IF NO REQUEST IN QUEUE
                                        ;
020422  005002                  GETREQ: CLR     R2
020424  105761  017756                  TSTB    QCNT(R1)                ;IS THERE ANY REQUEST IN QUEUE?
020430  001404                          BEQ     2$                      ;NO---BRANCH
020432  006301                  1$:     ASL     R1
020434  017102  020006                  MOV     @QOUTPT(R1),R2  ;PICKUP "DPB" POINTER FOR THIS DRIVE
020440  006201                          ASR     R1
020442  000207                  2$:     RTS     PC                      ;RETURN TO USER
                                        ;
                                        ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
                                        ;
                                        ;CALL
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
                                        ;       JSR     PC,POPQUE       ;CALL TO REMOVE REQUEST
                                        ;       RETURN                  ;R2=ADDRESS OF DPB REMOVED
                                        ;
020444  105361  017756          POPQUE: DECB    QCNT(R1)                ;DECREMENT QUEUE COUNT
020450  006301                          ASL     R1
020452  017102  020006                  MOV     @QOUTPT(R1),R2  ;GET THE "DPB" POINTER
020456  062761  000002  020006          ADD     #2,QOUTPT(R1)           ;UPDATE THE QUEUE POINTER
020464  026161  020006  020030          CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
020472  001003                          BNE     1$                      ;NO--BRANCH TO EXIT
020474  016161  020026  020006          MOV     QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
020502  006201                  1$:     ASR     R1
020504  000207                          RTS     PC                      ;RETURN TO USER
                                        ;
                                        ;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
                                        ;REPORTS AN ERROR DIRECTLY.
                                        ;
                                        ;CALL
                                        ;       JSR     R0,ES.SAV
                                        ;       ERROR   N                       ;;THE ERROR CALL
                                        ;       RETURN                  ;THE RETURN IS PAST THE ERROR CALL
                                        ;
```

```
                                ES.SAV: MOV     (R0)+,1$        ;GET THE ERROR CALL
                                        MOV     $ESCAPE,-(SP)   ;SAVE THE ADDRESS IN '$ESCAPE'
                                        CLR     $ESCAPE         ;CLEAR THE ESCAPE RETURN
                                1$:     .WORD   0               ;THE ERROR CALL IS MOVED HERE
                                        MOV     (SP)+,$ESCAPE   ;RESTORE THE ESCAPE ADDRESS
                                        RTS     R0              ;RETURN

                                ;;*****************************************************************

                                .SBTTL  TELETYPE MESSAGES

                                UNTOFF: .ASCIZ  / OFFLINE/
                                UNTON:  .ASCIZ  / ONLINE/
                                NOTRP:  .ASCIZ  @ NOT AN RP04/5/6@
                                NOTPRS: .ASCIZ  / NOT PRESENT/
                                NOTSAF: .ASCIZ  / UNSAFE/
                                RP04B:  .ASCIZ  /RP04/
                                RP05:   .ASCIZ  /RP05/
                                RP06:   .ASCIZ  /RP06/
                                SYSTAT: .ASCIZ  /UNIT STATUS/<CR><LF><LF>
                                MUNIT:  .ASCIZ  <CR><LF>/DRIVE: /
                                SLASH:  .ASCIZ  @ / @
                                C:      .ASCIZ  /C/
                                T:      .ASCIZ  /T/
                                MDRVD:  .ASCIZ  /D/
                                LIN4SP: .ASCII  /    /
                                LINSP:  .ASCIZ  /  /
                                ENTADR: .ASCIZ  /ENTER ADDRESS LIMITS:/<CR><LF>

                                MOFFLN: .ASCIZ  /  DRIVE OFFLINE/<CR><LF>

                                MDRNP:  .ASCIZ  / DRIVE NOT PRESENT/<CR><LF>

                                MER11:  .ASCIZ  / DRIVE NOT AVAILABLE/<CR><LF>

                                MNRP04: .ASCIZ  @ DRIVE NOT AN RP04/5/6@<CR><LF>
```

```
4400   021060   032457   033057   005015
4401   021066      000
4402   021067      040   051104   053111   MUSDR:   .ASCIZ  / DRIVE UNSAFE/<CR><LF>
4403   021074   020105   047125   040523
4404   021102   042506   005015      000
4405   021107      040   042523   042514   MSELD:   .ASCIZ  / SELECTED/
4406   021114   052103   042105      000
4407   021121      015   050012   047522   MMODE:   .ASCIZ  <CR><LF>/PROGRAM MODE (C OR F): /
4408   021126   051107   046501   046440
4409   021134   042117   020105   041450
4410   021142   047440   020122   024506
4411   021150   020072      000
4412   021153      040   047506   046522   MFORMT:  .ASCIZ  / FORMAT & VERIFY/
4413   021160   052101   023040   053040
4414   021166   051105   043111   000131
4415   021174   044103   041505   020113   MHECK:   .ASCIZ  /CHECK ONLY/
4416   021202   047117   054514      000
4417   021207      106   051117   040515   MORMAT:  .ASCIZ  /FORMAT & VERIFY/
4418   021214   020124   020046   042526
4419   021222   044522   054506      000
4420   021227      015   005012   050117   MSIZE:   .ASCIZ  <CR><LF><LF>/OPERATE IN 22 SECTOR MODE (Y OR N) ? /
4421   021234   051105   052101   020105
4422   021242   047111   031040   020062
4423   021250   042523   052103   051117
4424   021256   046440   042117   020105
4425   021264   054450   047440   020122
4426   021272   024516   037440   000040
4427   021300   050117   051105   052101   MSEC22:  .ASCIZ  /OPERATION WILL BE IN 22 SECTOR (16 BIT) MODE/<CR><LF>
4428   021306   047511   020116   044527
4429   021314   046114   041040   020105
4430   021322   047111   031040   020062
4431   021330   042523   052103   051117
4432   021336   024040   033061   041040
4433   021344   052111   020051   047515
4434   021352   042504   005015      000
4435   021357      117   042520   040522   MSEC20:  .ASCIZ  /OPERATION WILL BE IN 20 SECTOR (18 BIT) MODE/<CR><LF>
4436   021364   044524   047117   053440
4437   021372   046111   020114   042502
4438   021400   044440   020116   030062
4439   021406   051440   041505   047524
4440   021414   020122   030450   020070
4441   021422   044502   024524   046440
4442   021430   042117   006505   000012
4443   021436   047111   040526   044514   BADENT:  .ASCIZ  /INVALID ENTRY/<CR><LF>
4444   021444   020104   047105   051124
4445   021452   006531   000012
4446   021456   047105   044504   043516   MADRER:  .ASCII  /ENDING DSK ADRS MUST BE EQUAL TO OR GREATER/<CR><LF>
4447   021464   042040   045523   040440
4448   021472   051104   020123   052515
4449   021500   052123   041040   020105
4450   021506   050505   040525   020114
4451   021514   047524   047440   020122
4452   021522   051107   040505   042524
4453   021530   006522      012
4454   021533      124   040510   020116            .ASCIZ  /THAN STARTING ADRS/<CR><LF>
4455   021540   052123   051101   044524
```

```
4456  021546  043516  040440  051104
4457  021554  006523  000012
4458  021560  042523  042514  052103  MSELP:  .ASCII  /SELECT DATA PATTERN/<CR><LF>
4459  021566  042040  052101  020101
4460  021574  040520  052124  051105
4461  021602  006516     012
4462  021605     040  030050  020051          .ASCII  / (0) ZERO'S/<CR><LF>
4463  021612  042532  047522  051447
4464  021620  005015
4465  021622  024040  024461  047440          .ASCII  / (1) ONE'S/<CR><LF>
4466  021630  042516  051447  005015
4467  021636  024040  024462  053440          .ASCIZ  / (2) WORST CASE: /
4468  021644  051117  052123  041440
4469  021652  051501  035105  000040
4470
4471  021660  047527  051522  020124  MPATD:  .ASCIZ  /WORST CASE/
4472  021666  040503  042523     000
4473  021673     015  005012  052123  MSFOU:  .ASCIZ  <CR><LF><LF>/STARTING FORMAT ON DRIVE /
4474  021700  051101  044524  043516
4475  021706  043040  051117  040515
4476  021714  020124  047117  042040
4477  021722  044522  042526  000040
4478  021730  005015  051412  040524  MSCHK:  .ASCIZ  <CR><LF><LF>/STARTING CHECK ON DRIVE /
4479  021736  052122  047111  020107
4480  021744  044103  041505  020113
4481  021752  047117  042040  044522
4482  021760  042526  000040
4483  021764  005015  043012  051117  MFCMPT: .ASCIZ  <CR><LF><LF>/FORMAT COMPLETE, /
4484  021772  040515  020124  047503
4485  022000  050115  042514  042524
4486  022006  020054     000
4487  022011     015  005012  044103  MCCMPT: .ASCIZ  <CR><LF><LF>/CHECK COMPLETE, /
4488  022016  041505  020113  047503
4489  022024  050115  042514  042524
4490  022032  020054     000
4491  022035     124  052117  046101  NUMERR: .ASCIZ  /TOTAL ERRORS DETECTED: /
4492  022042  042440  051122  051117
4493  022050  020123  042504  042524
4494  022056  052103  042105  020072
4495  022064     000
4496  022065     120  042522  042523  ADDRIS: .ASCIZ  /PRESENT ADDRESS IS: /
4497  022072  052116  040440  042104
4498  022100  042522  051523  044440
4499  022106  035123  000040
4500
4501                                  ;;*****************************************************************
4502
4503                                  .SBTTL  ERROR MESSAGES
4504
4505                                  ;;*****************************************************************
4506
4507  022112  044122  030461  044440  EM1:    .ASCIZ  /RH11 INTERRUPT OCCURRED (RPAS=0)/
4508  022120  052116  051105  052522
4509  022126  052120  047440  041503
4510  022134  051125  042522  020104
4511  022142  051050  040520  036523
```

```
4512   022150   024460      000
4513   022153      125   042516   050130   EM2:    .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
4514   022160   041505   042524   020104
4515   022166   052101   042524   052116
4516   022174   047511   020116   041517
4517   022202   052503   051122   042105
4518   022210      000
4519   022211      115   051501   041123   EM3:    .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
4520   022216   051525   050040   051101
4521   022224   052111   020131   051105
4522   022232   047522   020122   046450
4523   022240   050103   036505   024461
4524   022246      000
4525   022247      115   051501   041123   EM4:    .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
4526   022254   051525   050040   051101
4527   022262   052111   020131   051105
4528   022270   047522   020122   050050
4529   022276   051101   030475   000051
4530   022304   042101   051104   051505   EM5:    .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
4531   022312   020123   046120   043525
4532   022320   041440   040510   043516
4533   022326   020105   044502   020124
4534   022334   042523   000124
4535   022340   044122   030461   042040   EM6:    .ASCIZ  /RH11 DIDN'T RESPOND TO ADDRESSING/
4536   022346   042111   023516   020124
4537   022354   042522   050123   047117
4538   022362   020104   047524   040440
4539   022370   042104   042522   051523
4540   022376   047111   000107
4541   022402   051104   053111   020105   EM10:   .ASCIZ  /DRIVE OFFLINE/
4542   022410   043117   046106   047111
4543   022416   000105
4544   022420   042520   051522   051511   EM11:   .ASCIZ  /PERSISTENT DRIVE UNSAFE ERROR/
4545   022426   042524   052116   042040
4546   022434   044522   042526   052440
4547   022442   051516   043101   020105
4548   022450   051105   047522   000122
4549   022456   047125   047503   051122   EM12:   .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
4550   022464   041505   040524   046102
4551   022472   020105   040515   051523
4552   022500   052502   020123   040520
4553   022506   044522   054524   042440
4554   022514   051122   051117      000
4555   022521      123   043117   053524   EM13:   .ASCIZ  /SOFTWARE TIMEOUT/
4556   022526   051101   020105   044524
4557   022534   042515   052517   000124
4558   022542   051104   053111   020105   EM14:   .ASCIZ  /DRIVE UNSAFE ERROR/
4559   022550   047125   040523   042506
4560   022556   042440   051122   051117
4561   022564      000
4562   022565      103   047117   051124   EM15:   .ASCIZ  @CONTROLLER/DRIVE ERROR DURING WRITE@
4563   022572   046117   042514   027522
4564   022600   051104   053111   020105
4565   022606   051105   047522   020122
4566   022614   052504   044522   043516
4567   022622   053440   044522   042524
```

```
4568  022630      000
4569  022631      103  047117  051124  EM16:   .ASCIZ  @CONTROLLER/DRIVE ERROR DURING WRITE CHECK@
4570  022636  046117  042514  027522
4571  022644  051104  053111  020105
4572  022652  051105  047522  020122
4573  022660  052504  044522  043516
4574  022666  053440  044522  042524
4575  022674  041440  042510  045503
4576  022702      000
4577  022703      122  052105  044522  EM17:   .ASCIZ  @RETRIES NOT SUCESSFUL - SECTOR NOT ACCEPTABLE@
4578  022710  051505  047040  052117
4579  022716  051440  041525  051505
4580  022724  043123  046125  026440
4581  022732  051440  041505  047524
4582  022740  020122  047516  020124
4583  022746  041501  042503  052120
4584  022754  041101  042514      000
4585  022761      104  052101  020101  EM20:   .ASCIZ  @DATA ERROR DURING WRITE CHECK@
4586  022766  051105  047522  020122
4587  022774  052504  044522  043516
4588  023002  053440  044522  042524
4589  023010  041440  042510  045503
4590  023016      000
4591  023017      103  047117  051124  EM21:   .ASCIZ  @CONTROLLER/DRIVE ERROR VERIFYING HEADERS@
4592  023024  046117  042514  027522
4593  023032  051104  053111  020105
4594  023040  051105  047522  020122
4595  023046  042526  044522  054506
4596  023054  047111  020107  042510
4597  023062  042101  051105  000123
4598  023070  042510  042101  051105  EM22:   .ASCIZ  @HEADER COMPARE ERROR VERIFYING HEADERS@
4599  023076  041440  046517  040520
4600  023104  042522  042440  051122
4601  023112  051117  053040  051105
4602  023120  043111  044531  043516
4603  023126  044040  040505  042504
4604  023134  051522      000
4605  023137      103  046131  047111  EM23:   .ASCIZ  @CYLINDER FIELD IN HEADER IS NOT CORRECT@
4606  023144  042504  020122  044506
4607  023152  046105  020104  047111
4608  023160  044040  040505  042504
4609  023166  020122  051511  047040
4610  023174  052117  041440  051117
4611  023202  042522  052103      000
4612  023207      127  044522  042524  EM24:   .ASCIZ  @WRITE CHECK ERROR@
4613  023214  041440  042510  045503
4614  023222  042440  051122  051117
4615  023230      000
4616
4617                                  ;;*****************************************************************
4618
4619  023231      122  040520  000123  DH1:    .ASCIZ  /RPAS/
4620  023236  051104  053111  020105  DH2:    .ASCIZ  /DRIVE   RPDS1   RPER1   RPER2   RPER3   RPAS/
4621  023244  020040  050122  051504
4622  023252  020061  020040  050122
4623  023260  051105  020061  020040
```

```
4624   023266   050122   051105   020062
4625   023274   020040   050122   051105
4626   023302   020063   020040   050122
4627   023310   051501      000
4628   023313      104   044522   042526   DH3:    .ASCIZ  /DRIVE   REG ADR  DATA/
4629   023320   020040   051040   043505
4630   023326   040440   051104   020040
4631   023334   040504   040524      000
4632   023341      104   044522   042526   DH4:    .ASCIZ  /DRIVE   REG ADR    GOOD    BAD/
4633   023346   020040   051040   043505
4634   023354   040440   051104   020040
4635   023362   020040   047507   042117
4636   023370   020040   020040   040502
4637   023376   000104
4638   023400   044122   040440   042104   DH6:    .ASCIZ  /RH ADDRESS/
4639   023406   042522   051523      000
4640   023413      104   044522   042526   DH10:   .ASCIZ  /DRIVE   ERR PC  RPCS1   RPCS2   RPDS1   RPER1   RPER2   RPER3/
4641   023420   020040   042440   051122
4642   023426   050040   020103   051040
4643   023434   041520   030523   020040
4644   023442   051040   041520   031123
4645   023450   020040   051040   042120
4646   023456   030523   020040   051040
4647   023464   042520   030522   020040
4648   023472   051040   042520   031122
4649   023500   020040   051040   042520
4650   023506   031522      000
4651   023511      122   042520   030503   DH10A:  .ASCIZ  /RPEC1   RPEC2   RPWC    RPBA    RPDA    RPAS    RPLA/
4652   023516   020040   051040   042520
4653   023524   031103   020040   051040
4654   023532   053520   020103   020040
4655   023540   051040   041120   020101
4656   023546   020040   051040   042120
4657   023554   020101   020040   051040
4658   023562   040520   020123   020040
4659   023570   051040   046120   000101
4660   023576   050122   041104   020040   DH10B:  .ASCIZ  /RPDB    RPMR    RPDT    RPSN    RPOF    RPCA    RPCC/
4661   023604   020040   050122   051115
4662   023612   020040   020040   050122
4663   023620   052104   020040   020040
4664   023626   050122   047123   020040
4665   023634   020040   050122   043117
4666   023642   020040   020040   050122
4667   023650   040503   020040   020040
4668   023656   050122   041503      000
4669   023663      104   044522   042526   DH17:   .ASCIZ  /DRIVE   ERR PC CYLINDER TRACK   SECTOR/
4670   023670   020040   042440   051122
4671   023676   050040   020103   054503
4672   023704   044514   042116   051105
4673   023712   052040   040522   045503
4674   023720   020040   051440   041505
4675   023726   047524   000122
4676   023732   051104   053111   020105   DH20:   .ASCIZ  /DRIVE   ERR PC CYLINDER TRACK   SECTOR/
4677   023740   020040   051105   020122
4678   023746   041520   041440   046131
4679   023754   047111   042504   020122
```

```
 4680  023762  051124  041501  020113
 4681  023770  020040  042523  052103
 4682  023776  051117     000
 4683  024001     122  041520  030523  DH20A:  .ASCIZ  /RPCS1   RPCS2   RPDS1   RPER1   RPER2   RPER3   RPEC1   RPEC2/
 4684  024006  020040  051040  041520
 4685  024014  031123  020040  051040
 4686  024022  042120  030523  020040
 4687  024030  051040  042520  030522
 4688  024036  020040  051040  042520
 4689  024044  031122  020040  051040
 4690  024052  042520  031522  020040
 4691  024060  051040  042520  030503
 4692  024066  020040  051040  042520
 4693  024074  031103     000
 4694  024077     122  053520  020103  DH20B:  .ASCIZ  /RPWC    RPBA    RPDA    RPAS    RPLA    RPDB    RPMR    RPDT/
 4695  024104  020040  051040  041120
 4696  024112  020101  020040  051040
 4697  024120  042120  020101  020040
 4698  024126  051040  040520  020123
 4699  024134  020040  051040  046120
 4700  024142  020101  020040  051040
 4701  024150  042120  020102  020040
 4702  024156  051040  046520  020122
 4703  024164  020040  051040  042120
 4704  024172  000124
 4705  024174  050122  047123  020040  DH20C:  .ASCIZ  /RPSN    RPOF    RPCA    RPCC/
 4706  024202  020040  050122  043117
 4707  024210  020040  020040  050122
 4708  024216  040503  020040  020040
 4709  024224  050122  041503     000
 4710  024231     040  020040  020040  DH23:   .ASCII  /               EXPT'D/<CR><LF>
 4711  024236  020040  020040  020040
 4712  024244  020040  020040  042440
 4713  024252  050130  023524  006504
 4714  024260     012
 4715  024261     104  044522  042526          .ASCIZ  /DRIVE   ERR PC  CYLNDR  ACTUAL HEADER/
 4716  024266  020040  042440  051122
 4717  024274  050040  020103  041440
 4718  024302  046131  042116  020122
 4719  024310  040440  052103  040525
 4720  024316  020114  042510  042101
 4721  024324  051105     000
 4722  024327     040  020040  020040  DH24:   .ASCII  /                                       MEMORY  DISK/<CR><LF>
 4723  024334  020040  020040  020040
 4724  024342  020040  020040  020040
 4725  024350  020040  020040  020040
 4726  024356  020040  020040  020040
 4727  024364  020040  020040  020040
 4728  024372  020040  020040  046440
 4729  024400  046505  051117  020131
 4730  024406  042040  051511  006513
 4731  024414     012
 4732  024415     104  044522  042526          .ASCIZ  /DRIVE   ERR PC  CYLNDR  TRACK   SECTOR  DATA    DATA/
 4733  024422  020040  042440  051122
 4734  024430  050040  020103  041440
 4735  024436  046131  042116  020122
```

```
 4736  024444  052040  040522  045503
 4737  024452  020040  051440  041505
 4738  024460  047524  020122  042040
 4739  024466  052101  020101  020040
 4740  024474  042040  052101  000101
 4741
 4742
 4743  024502  001276                  DT1:    .WORD   ATTN
 4744  024504  001274  012100  012102  DT2:    .WORD   DDRIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN
 4745  024512  012104  012106  001276
 4746  024520  001274  017240  017242  DT3:    .WORD   DDRIVE,RD.ADR,RD.WRD
 4747  024526  001274  017464  017462  DT4:    .WORD   DDRIVE,WRT.AD,WRT.WD,RD.WRD
 4748  024534  017242
 4749  024536  001172                  DT6:    .WORD   $RPADR
 4750  024540  001214  001116  001304  DT10:   .WORD   DRIVE,$ERRPC,RP.REG,RP.REG+10,RP.REG+12,RP.REG+14,RP.REG+40,RP.REG+42
 4751  024546  001314  001316  001320
 4752  024554  001344  001346
 4753  024560  001350  001352  001306          .WORD   RP.REG+44,RP.REG+46,RP.REG+2,RP.REG+4,RP.REG+6,RP.REG+16,RP.REG+20
 4754  024566  001310  001312  001322
 4755  024574  001324
 4756  024576  001326  001330  001332          .WORD   RP.REG+22,RP.REG+24,RP.REG+26,RP.REG+30,RP.REG+32,RP.REG+34,RP.REG+36
 4757  024604  001334  001336  001340
 4758  024612  001342
 4759  024614  001214  001116  001270  DT17:   .WORD   DRIVE,$ERRPC,DS.CYL,DS.TRK,SAVSEC
 4760  024622  001272  001252
 4761  024626  001214  001116  001270  DT20:   .WORD   DRIVE,$ERRPC,DS.CYL,DS.TRK,SAVSEC
 4762  024634  001272  001252
 4763  024640  001304  001314  001316          .WORD   RP.REG,RP.REG+10,RP.REG+12,RP.REG+14,RP.REG+40,RP.REG+42,RP.REG+44,RP.RE
 4764  024646  001320  001344  001346
 4765  024654  001350  001352
 4766  024660  001306  001310  001312          .WORD   RP.REG+2,RP.REG+4,RP.REG+6,RP.REG+16,RP.REG+20,RP.REG+22,RP.REG+24,RP.RE
 4767  024666  001322  001324  001326
 4768  024674  001330  001332
 4769  024700  001334  001336  001340          .WORD   RP.REG+30,RP.REG+32,RP.REG+34,RP.REG+36
 4770  024706  001342
 4771  024710  001214  001116  025130  DT23:   .WORD   DRIVE,$ERRPC,BUFP,RBUF,RBUF+2,RBUF+4,RBUF+6
 4772  024716  025120  025122  025124
 4773  024724  025126
 4774  024726  001214  001116  001270  DT24:   .WORD   DRIVE,$ERRPC,DS.CYL,DS.TRK,SAVSEC,$GDDAT,RP.REG+22
 4775  024734  001272  001252  001124
 4776  024742  001326
 4777  024744  001304  001314  001316          .WORD   RP.REG,RP.REG+10,RP.REG+12,RP.REG+14,RP.REG+40,RP.REG+42,RP.REG+44,RP.RE
 4778  024752  001320  001344  001346
 4779  024760  001350  001352
 4780  024764  001306  001310  001312          .WORD   RP.REG+2,RP.REG+4,RP.REG+6,RP.REG+16,RP.REG+20,RP.REG+22,RP.REG+24,RP.RE
 4781  024772  001322  001324  001326
 4782  025000  001330  001332
 4783  025004  001334  001336  001340          .WORD   RP.REG+30,RP.REG+32,RP.REG+34,RP.REG+36
 4784  025012  001342
 4785
 4786  025014  000001                  DF1:    .WORD   1
 4787  025016     001     000                  .BYTE   1,0
 4788  025020  000001                  DF2:    .WORD   1
 4789  025022     006     000                  .BYTE   6,0
 4790  025024  000001                  DF3:    .WORD   1
 4791  025026     003     000                  .BYTE   3,0
```

```
4792  025030  000001                DF4:    .WORD   1
4793  025032     004      000               .BYTE   4,0
4794  025034  000003                DF10:   .WORD   3
4795  025036     010      000               .BYTE   8.,0
4796  025040  023511                        .WORD   DH10A
4797  025042     007      000               .BYTE   7,0
4798  025044  023576                        .WORD   DH10B
4799  025046     007      000               .BYTE   7,0
4800  025050  000001                DF17:   .WORD   1
4801  025052     005      034               .BYTE   5,34
4802  025054  000004                DF20:   .WORD   4
4803  025056     005      034               .BYTE   5,34
4804  025060  024001                        .WORD   DH20A
4805  025062     010      000               .BYTE   8.,0
4806  025064  024077                        .WORD   DH20B
4807  025066     010      000               .BYTE   8.,0
4808  025070  024174                        .WORD   DH20C
4809  025072     004      000               .BYTE   4,0
4810  025074  000001                DF23:   .WORD   1
4811  025076     007      000               .BYTE   7,0
4812  025100  000004                DF24:   .WORD   4
4813  025102     007      034               .BYTE   7,34
4814  025104  024001                        .WORD   DH20A
4815  025106     010      000               .BYTE   8.,0
4816  025110  024077                        .WORD   DH20B
4817  025112     010      000               .BYTE   8.,0
4818  025114  024174                        .WORD   DH20C
4819  025116     004      000               .BYTE   4,0
4820
4821                                ;;*****************************************************************
4822
4823                                ;BUFFER STARTS HERE
4824
4825                                ;;*****************************************************************
4826
4827  025120  000000  000000  000000  RBUF:   .WORD   0,0,0,0          ;BUFFER FOR HEADER CHECK
4828  025126  000000
4829
4830  025130                          BUFP:                   ;FORMAT AND CHECK BUFFER STARTS HERE
4831
4832  025130  005015  055012  026532  TITLE:  .ASCII  <CR><LF><LF>/ZZ-CZRJB-B/<CR><LF>
4833  025136  055103  045122  026502
4834  025144  006502     012
4835  025147     122  030120  027464          .ASCIZ  @RP04/5/6 FORMATTER PROGRAM@<CR><LF><LF>
4836  025154  027465  020066  047506
4837  025162  046522  052101  042524
4838  025170  020122  051120  043517
4839  025176  040522  006515  005012
4840  025204     000
4841  025205     015  052012  020117  LOADRV: .ASCII  <CR><LF>/TO 'FORMAT' OR 'CHECK' DRIVE 0, REPLACE THE 'XXDP'/<CR><LF>
4842  025212  043047  051117  040515
4843  025220  023524  047440  020122
4844  025226  041447  042510  045503
4845  025234  020047  051104  053111
4846  025242  020105  026060  051040
4847  025250  050105  040514  042503
```

```
 4848  025256  052040  042510  023440
 4849  025264  054130  050104  006447
 4850  025272     012
 4851  025273     120  041501  020113          .ASCII  /PACK ON DRIVE 0 WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,/<CR><LF>
 4852  025300  047117  042040  044522
 4853  025306  042526  030040  053440
 4854  025314  052111  020110  047101
 4855  025322  052117  042510  020122
 4856  025330  040520  045503  020054
 4857  025336  046103  040505  020122
 4858  025344  042515  047515  054522
 4859  025352  046040  041517  052101
 4860  025360  047511  020116  030064
 4861  025366  006454     012
 4862  025371     101  042116  051040          .ASCIZ  /AND RESTART THE PROGRAM/<CR><LF>
 4863  025376  051505  040524  052122
 4864  025404  052040  042510  050040
 4865  025412  047522  051107  046501
 4866  025420  005015     000
 4867
 4868
 4869                                  .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
 4870                                  ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
 4871                                  ;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
 4872                                  ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
 4873                                  ;REQUIRED.
 4874                                  ;NOTE: THIS ROUTINE DESTROYS R0-R4
 4875                                  ;CALL
 4876                                  ;
 4877                                  ;       JSR     PC,BUSADR
 4878                                  ;       RETURN
 4879
 4880  025424  177700                  HIAD:   .WORD 177700
 4881  025426  000776                  HIVEC:  .WORD 776
 4882  025430  000000                  LIMIT:  .WORD
 4883  025432  005737  001302          BUSADR: TST     CHGADR          ;INPUT FROM TTY REQUESTED?
 4884  025436  001460                          BEQ     7$              ;NO--BRANCH
 4885  025440  005037  001302                  CLR     CHGADR          ;YES--CLEAR THE REQUEST FLAG
 4886  025444  012700  001172          1$:     MOV     #$RPADR,R0      ;FIRST ADDRESS
 4887  025450  104401  025652                  TYPE    ,MRPCS1         ;"RPCS1="
 4888  025454  012046                          MOV     (R0)+,-(SP)     ;PRESENT RPCS1 ADDRESS
 4889  025456  104402                          TYPOC                   ;TYPE IT
 4890  025460  104401  020705                  TYPE    ,LINSP          ;2 SPACES
 4891  025464  104411                          RDLIN                   ;GET THE ENTRY
 4892  025466  012601                          MOV     (SP)+,R1        ;ADDRESS OF ASCII TEXT
 4893  025470  013737  025424  025430          MOV     HIAD,LIMIT      ;THIS IS THE ADDRSS HIGH LIMIT
 4894  025476  004537  025674                  JSR     R5,CK.NUM       ;CHECK THE NUMBER
 4895  025502  025522                          3$                      ;CARRIAGE RETURN ONLY ENTERED
 4896  025504  025600                          7$                      ;PERIOD ONLY ENTERED
 4897  025506  025444                          1$                      ;ILLEGAL INPUT
 4898  025510  025516                          2$                      ;TERMINATED WITH A CARRIAGE RETURN
 4899  025512  025444                          1$                      ;TERMINATED WITH A "."
 4900  025514  025574                          4$                      ;TERMINATED WITH A "'."
 4901  025516  010260  177776          2$:     MOV     R2,-2(R0)       ;SAVE NEW RPCS1
 4902  025522  104401  025663          3$:     TYPE    ,MRHVEC         ;"RHVEC="
 4903  025526  012700  001174                  MOV     #$RPVEC,R0      ;FIRST VECTOR
```

```
                                        MOV     (R0)+,-(SP)     ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
                                        TYPOC                   ;TYPE IT
                                        TYPE    ,LINSP          ;2 SPACES
                                        RDLIN                   ;READ THE ENTRY
                                        MOV     (SP)+,R1        ;ASCII TEXT ADDRESS
                                        MOV     HIVEC,LIMIT     ;VECTOR LIMIT
                                        JSR     R5,CK.NUM       ;CHECK THE NUMBER
                                        7$                      ;CARRIAGE RETURN ONLY ENTERED
                                        7$                      ;PERIOD ONLY ENTERED
                                        3$                      ;ILLEGAL INPUT
                                        4$                      ;TERMINATED WITH A CARRIAGE RETURN
                                        3$                      ;TERMINATED WITH A ","
                                        4$                      ;TERMINATED WITH A "."
                                4$:     MOV     R2,-2(R0)       ;SAVE INPUT
                                7$:     MOV     ERRVEC,R1       ;SAVE THE ERROR VECTOR
                                        MOV     #8$,ERRVEC      ;SETUP FOR TRAP
                                        TST     @$RPADR         ;CHECK FOR RH11
                                        MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
                                        MOV     #$RPADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
                                        MOV     #RPADR,R1       ;FIRST ADDRESS OF WHERE TO PUT THEM
                                        MOV     (R0)+,(R1)+     ;BUS ADDRESS
                                        MOV     (R0)+,(R1)+     ;VECTOR ADDRESS
                                        RTS     PC              ;RETURN
                                8$:     MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
                                        CMP     (SP)+,(SP)+     ;CLEAN OFF THE STACK
                                        ERROR   6               ;REPORT THE ERROR
                                        BR      1$              ;ASK FOR BUS ADDRESS

MRPCS1: .ASCIZ  @RPCS1 = @
MRHVEC: .ASCIZ  @RHVEC = @


;SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
;AND FORMS AN OCTAL NUMBER IN R2
;CALL:
;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
;       MOV     #NUM,R2         ;MAX SIZE OF INPUT NUMBER
;       JSR     R5,CK.NUM       ;GO FORM THE NUMBER
;       RETURN  ADR1            ;"CR" ONLY ENTERED -- R2 = 0
;       RETURN  ADR2            ;"PERIOD" ONLY ENTERED -- R2 = 0
;       RETURN  ADR3            ;ILLEGAL CHARACTER IN THE INPUT STRING
;       RETURN  ADR4            ;"CR" ENTERED -- R2 = NUMBER
;       RETURN  ADR5            ;"COMMA" -- R2 = NUMBER
;       RETURN  ADR6            ;"PERIOD" -- R2 = NUMBER

CK.NUM: MOV     R4,-(SP)        ;SAVE R4
        MOV     R3,-(SP)        ;SAVE R3
        MOV     R2,-(SP)        ;SAVE R2
        CLR     R4              ;RETURN POINTER
        CLR     R3              ;START NUMBER AT ZERO
        CLR     R2              ;STORE RESULT
        JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
        6$                      ;ILLEGAL CHARACTER
        8$                      ;CARRIAGE RETURN
```

```
 4960  025720  026014                          6$                      ;"."
 4961  025722  026016                          7$                      ;"-"
 4962  025724  025730                          1$                      ;DIGIT 0-7
 4963  025726  026014                          6$                      ;DIGIT 8-9
 4964  025730  062705  000004        1$:       ADD     #4,R5           ;INCREMENT RETURN PAST "CR" AND "PERIOD" ONLY RETURNS
 4965  025734  006303                2$:       ASL     R3              ;FOR THE OCTAL NUMBER IN R3
 4966  025736  103426                          BCS     6$              ;DON'T LET IT GET TO BIG
 4967  025740  006303                          ASL     R3
 4968  025742  103424                          BCS     6$
 4969  025744  006303                          ASL     R3
 4970  025746  103422                          BCS     6$
 4971  025750  060203                          ADD     R2,R3
 4972  025752  004537  006426                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
 4973  025756  026020                          8$                      ;ILLEGAL CHARACTER
 4974  025760  026002                          5$                      ;CARRIAGE RETURN
 4975  025762  026000                          4$                      ;"."
 4976  025764  025772                          3$                      ;"-"
 4977  025766  025734                          2$                      ;DIGIT 0-7
 4978  025770  026020                          8$                      ;DIGIT 8-9
 4979  025772  105711                3$:       TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
 4980  025774  001011                          BNE     8$              ;BR IF NOT
 4981  025776  005724                          TST     (R4)+           ;INCREMENT THE RETURN
 4982  026000  005724                4$:       TST     (R4)+           ;INCREMENT THE RETURN INDEX
 4983  026002  005724                5$:       TST     (R4)+           ;INCREMENT THE RETURN INDEX
 4984  026004  023703  025430                  CMP     LIMIT,R3        ;INPUT VALUE TOO LARGE ?
 4985  026010  101003                          BHI     8$              ;BR IF IT IS
 4986  026012  000401                          BR      7$              ;BR IF NOT
 4987  026014  005725                6$:       TST     (R5)+           ;INCREMENT THE RETURN ADDRESS
 4988  026016  005725                7$:       TST     (R5)+           ;INCREMENT THE RETURN ADDRESS
 4989  026020  060405                8$:       ADD     R4,R5           ;SETUP FOR PROPER RETURN
 4990  026022  010302                          MOV     R3,R2           ;LOAD ENTERED VALUE
 4991  026024  005726                          TST     (SP)+           ;CLEAN OFF THE STACK
 4992  026026  012603                          MOV     (SP)+,R3        ;RESTORE R3
 4993  026030  012604                          MOV     (SP)+,R4        ;RESTORE R4
 4994  026032  011505                          MOV     (R5),R5         ;GET RETURN ADDRESS
 4995  026034  000205                          RTS     R5              ;RETURN
 4996
 4997
 4998
 4999  000001                          .END
```

```
ABS    = 000200
ACK    = 000123
ACL    = 000040
ACTDRV  012164
ACTSTR  012165
ACU    = 100000
ADDRIS  022065
ADRTBL  001476
AOE    = 001000
ATA    = 100000
ATABIT  012234
ATTN    001276
ATO    = 000001
AT1    = 000002
AT2    = 000004
AT3    = 000010
AT4    = 000020
AT5    = 000040
AT6    = 000100
AT7    = 000200
A16    = 000400
A17    = 001000
BADENT  021436
BAI    = 000010
BEGCYL  001224
BEGIN   002066
BEGIN1  002076
BEGIN2  002102
BEGTRK  001230
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BPTVEC= 000014
BUFP    025130
BUSADR  025432
C       020675
CHGADR  001302
CI1     013606
CI3     013714
CI4     014022
CI5     014400
CI6     014422
CI7     014436
CI7B    014464
CI8     014536
CKADRS  003570
CKSWR  = 104407
CKTRK   004236
CK.CHR  006426
CK.DEC  006400
CK.DIG  006500
CK.NUM  025674
CK.OCT  006352
CLOCK   006104
CLR    = 000040
CLRQUE  020250
CNTLC   001300
CR     = 000015
CRLF   = 000200
CSF    = 000002
CSU    = 000010
CYLCK   001246
DCK    = 100000
DCL    = 000100
DCU    = 000001
DDISP  = 177570
DDRIVE  001274
DE1    = 000040
DFF20  = 000002
DF1     025014
DF10    025034
DF17    025050
DF2     025020
DF20    025054
DF23    025074
DF24    025100
DF3     025024
DF4     025030
DH1     023231
DH10    023413
DH10A   023511
DH10B   023576
DH17    023663
DH2     023236
DH20    023732
DH20A   024001
DH20B   024077
DH20C   024174
DH23    024231
DH24    024327
DH3     023313
DH4     023341
DH6     023400
DIGB   = 000004
DISPLA  001142
DISPRE  000174
DLT    = 100000
DL64   = 000020
DMD    = 000001
DONE    005210
DPINT   012140
DPR    = 000400
DPRQS   012150
DRIVE   001214
DRQ    = 004000
DRVACT  012110
DRVCLR= 000111
DRVINT  012476
DRVQUE  020346
DRVSTA  012120
DRVTYP  012130
DRY    = 000200
DSWR   = 177570
DS.CYL  001270
DS.TRK  001272
DTE    = 010000
DTSY   = 000200
DTUW    012232
DT00   = 000001
DT01   = 000002
DT02   = 000004
DT03   = 000010
DT04   = 000020
DT05   = 000040
DT06   = 000100
DT07   = 000200
DT08   = 000400
DT1     024502
DT10    024540
DT17    024614
DT2     024504
DT20    024626
DT23    024710
DT24    024726
DT3     024520
DT4     024526
DT6     024536
DVA    = 004000
ECH    = 000100
ECI    = 004000
EMPTYQ  020326
EMTVEC= 000030
EM1     022112
EM10    022402
EM11    022420
EM12    022456
EM13    022521
EM14    022542
EM15    022565
EM16    022631
EM17    022703
EM2     022153
EM20    022761
EM21    023017
EM22    023070
EM23    023137
EM24    023207
EM3     022211
EM4     022247
EM5     022304
EM6     022340
ENDCYL  001222
ENDTRK  001226
ENTADR  020710
ERINDX  005234
ERR    = 040000
ERRVEC= 000004
ES.SAV  020506
EXT1   = 000001
EXT10  = 000010
EXT2   = 000002
EXT20  = 000020
EXT4   = 000004
EXT40  = 000040
FEN    = 000200
FER    = 000020
FMTDPB  001354
FMT22  = 010000
F1     = 000002
F2     = 000004
F3     = 000010
F4     = 000020
F5     = 000040
GETREG= 000141
GETREQ  020422
GO     = 000001
GRV    = 000010
GTSWR  = 104406
HCE    = 000200
HCI    = 002000
HCRC   = 000400
HDREAD  004650
HEDERR  001262
HIAD    025424
HIVEC   025426
HT     = 000011
IAE    = 002000
IE     = 000100
ILF    = 000001
ILR    = 000002
IOTVEC= 000020
IR     = 000100
ISR     015130
IXE    = 004000
LA      014772
LACNT   012176
LF     = 000012
LIMIT   025430
LINSP   020705
LIN4SP  020703
LOADRV  025205
LOP.CK  005334
LST    = 002000
MADRER  021456
MAXSEC  001266
MCCMPT  022011
MCLK   = 000002
MCPE   = 020000
MCPEMX  012244
MDRNP   020762
MDRVD   020701
MER11   021007
MFCMPT  021764
MFORMT  021153
MHECK   021174
MHS    = 001000
MINX   = 000004
MMODE   021121
MNDLTA  012260
MNRP04  021036
MODE    001220
MOFFLN  020740
MOH    = 020000
MOL    = 010000
MORMAT  021207
MPATD   021660
MPE    = 000400
MRD    = 000020
MRHVEC  025663
MRPCS1  025652
MSCHK   021730
MSE    = 000020
```

```
MSEC20  021357
MSEC22  021300
MSELD   021107
MSELP   021560
MSFOU   021673
MSIZE   021227
MSTCK = 000010
MUNIT   020657
MUSDR   021067
MWC     001260
MWR   = 000040
MXDLTA  012256
MXF   = 001000
MXLACT  012254
MXWNDW  012262
M0      003116
M1      002644
M1A     002734
M1B     003062
M2      003404
M4      003444
M5      003526
NBA   = 100000
NED   = 010000
NEM   = 004000
NHS   = 002000
NOTPRS  020574
NOTRP   020553
NOTSAF  020611
NUMERR  022035
OCYL  = 100000
OENTER  006116
OFFSET= 000115
OFREV = 000200
OF100 = 000004
OF200 = 000010
OF25  = 000001
OF400 = 000020
OF50  = 000002
OF800 = 000040
OPE   = 020000
OPI   = 020000
OPT     013326
OR    = 000200
PAR   = 000010
PARENT  006236
PAR1    001426
PAR2    001441
PAR3    001454
PAR4    001465
PAT   = 000020
PATA    001242
PATB    001244
PATSEL  001240
PGE   = 002000
PGM   = 001000
PIP   = 020000
PIRQ  = 177772
PIRQVE= 000240
PLU   = 020000
POPQUE  020444
PR0   = 000000
PR1   = 000040
PR2   = 000100
PR3   = 000140
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSEL  = 002000
PSU   = 000001
PSW   = 177776
PWRVEC= 000024
QCNT    017756
QDRV0   020050
QDRV1   020070
QDRV2   020110
QDRV3   020130
QDRV4   020150
QDRV5   020170
QDRV6   020210
QDRV7   020230
QINPT   017766
QOUTPT  020006
QSTART  020026
QSTOP   020030
QTERM = 020250
RAW   = 000020
RBUF    025120
RDCHR = 104410
RDDAT = 000171
RDHD  = 000173
RDLIN = 104411
RDY   = 000200
RD.ADR  017240
RD.RP   017214
RD.RP1  017236
RD.RP2  017362
RD.RP3  017366
RD.RP4  017372
RD.WRD  017242
READIN= 000121
RECAL = 000107
RELSE = 000113
RESREG= 104413
RESVEC= 000010
RETRY   001250
RMR   = 000004
RNOP  = 000101
RPADR   012246
RPAS  = 000016
RPBA  = 000004
RPCA  = 000034
RPCC  = 000036
RPCS1 = 000000
RPCS2 = 000010
RPDA  = 000006
RPDB  = 000022
RPDS1 = 000012
RPDT  = 000026
RPEC1 = 000044
RPEC2 = 000046
RPERRS  012100
RPER1 = 000014
RPER2 = 000040
RPER3 = 000042
RPINIT  012264
RPLA  = 000020
RPMR  = 000024
RPOF  = 000032
RPSN  = 000030
RPTMR   016466
RPVEC   012250
RPWC  = 000002
RP.REG  001304
RP04    013034
RP04B   020621
RP05    020626
RP06    020633
RTC   = 000117
R6    =%000006
R7    =%000007
SAVEFG  012206
SAVREG= 104412
SAVSEC  001252
SAVWC   001254
SC      015334
SCAWC   005554
SC1   = 000100
SC10  = 001000
SC11    016166
SC12    016256
SC13    016326
SC2   = 000200
SC20  = 002000
SC3     015404
SC4     015410
SC5     015422
SC6     015622
SC6A    015732
SC7     016060
SC8     016136
SEARCH= 000131
SEC20   001264
SEEK  = 000105
SEEKFG  012210
SELDRV= 000145
SETFMT= 000143
SETHDR  005602
SETPAT  003700
SETTBL  005402
SETVEC  002410
SET.IE  017704
SKI   = 040000
SLASH   020671
SOFSW   001216
SRCHWT  012162
STACK = 001100
STCLK1  006020
STCLK2  006070
STCLK3  006074
STKLMT= 177774
STO     016560
STO1    016610
STO2    017006
STO3    017056
STO5    017102
STO6    017110
STO7    017146
STO8    017176
STO9    017206
ST.CLK  005742
SVRH11  017566
SWR     001140
SWREG   000176
SW0   = 000001
SW00  = 000001
SW01  = 000002
SW02  = 000004
SW03  = 000010
SW04  = 000020
SW05  = 000040
SW06  = 000100
SW07  = 000200
SW08  = 000400
SW09  = 001000
SW1   = 000002
SW10  = 002000
SW11  = 004000
SW12  = 010000
SW13  = 020000
SW14  = 040000
SW15  = 100000
SW2   = 000004
SW3   = 000010
SW4   = 000020
SW5   = 000040
SW6   = 000100
SW7   = 000200
SW8   = 000400
SW9   = 001000
SYSTAT  020640
T       020677
TABLE   001374
TAP   = 040000
TBITVE= 000014
TD      015174
TDF   = 000040
TIMER   012212
TITLE   025130
TKVEC = 000060
TPVEC = 000064
TRAPVE= 000034
TRE   = 040000
TRKCNT  001236
TRKTST  005472
TRK1  = 004000
TRK10 = 040000
TRK2  = 010000
TRK20 = 100000
TRK4  = 020000
TRNSWT  012160
TRTVEC= 000014
TTRKS   001232
TTRKSC  001234
TUF   = 000100
TYPADR  006136
TYPDS = 104405
TYPE  = 104401
TYPERR  007044
TYPOC = 104402
TYPON = 104404
TYPOS = 104403
ULOFLG  012166
UNLOAD= 000103
UNS   = 040000
UNTOFF  020532
UNTON   020543
UPDACY  005662
UPDATK  005712
UPE   = 020000
US1   = 000001
US2   = 000002
US4   = 000004
```

```
UWR    = 000010        WRT.WD  017462        $ERROR  006636        $OCNT   007736        $TKSRV  010254
VUF    = 000002        WRU    = 000400       $ERRPC  001116        $OMODE  007740        $TN    = 000000
VU30   = 010000        WSU    = 000004       $ERRTB  001626        $PASS   001100        $TNPWR  011634
VV     = 000100        $AUTOB  001134        $ERTTL  001112        $QUES   001166        $TPB    001152
WAO    = 000002        $BDADR  001122        $ESCAP  001160        $RDCHR  011034        $TPFLG  001157
WC       001256        $BDDAT  001126        $FILLC  001156        $RDLIN  011124        $TPS    001150
WCE    = 040000        $BELL   001162        $FILLS  001155        $RDSZ  = 000007        $TRAP   012014
WCF    = 000040        $CHARC  007510        $GDADR  001120        $RESRE  011756        $TRAP2  012036
WCKD   = 000151        $CKSWR  010472        $GDDAT  001124        $RPADR  001172        $TRP   = 000014
WCKHD  = 000153        $CMTAG  001100        $GET42  005164        $RPVEC  001174        $TRPAD  012050
WCTBL    001552        $CM3   = 000000       $GTSWR  010562        $RTNAD  005206        $TSTNM  001102
WCU    = 000001        $CNTLC  011367        $HD    = 000000       $SAVRE  011720        $TTYIN  011360
WLE    = 004000        $CNTLG  011401        $ICNT   001104        $SB2D   011470        $TYPDS  007742
WRL    = 004000        $CNTLU  011374        $INTAG  001135        $SETUP= 000106        $TYPE   007274
WRTDAT= 000161         $CRLF   001167        $ITEMB  001114        $STUP  = 177777        $TYPEC  007444
WRTHD  = 000163        $DBLK   010156        $LF     001170        $SUPRS  011430        $TYPEX  007512
WRTRK    004004        $DB2D   011524        $LKCSB  001200        $SVPC  = 000200        $TYPOC  007540
WRTRK1   004060        $DECVL  011704        $LKCSR  001176        $SWR   = 123000        $TYPON  007554
WRTRK2   004116        $DOAGN  005204        $LKS    001206        $TKB    001146        $TYPOS  007514
WRT.AD   017464        $DTBL   010146        $LLVEC  001210        $TKCNT  010166        $$GET4= 000000
WRT.RP   017374        $ENDAD  005174        $LPADR  001106        $TKINT  010204        $OFILL  007737
WRT.R1   017460        $ENDCT  005160        $LPERR  001110        $TKQEN= 010203        .      = 026036
WRT.R2   017550        $EOP    005100        $LPVEC  001202        $TKQIN  010170
WRT.R3   017556        $EOPCT  005152        $MNEW   011417        $TKQOU  010172
WRT.R4   017562        $ERFLG  001103        $MSWR   011406        $TKQSR  010174
WRT.R5   017564        $ERMAX  001115        $NULL   001154        $TKS    001144

. ABS.   026036      000

 ERRORS DETECTED:  0

 RM03:CZRJBB,CZRJBB.SEQ/SOL/LI:ME/NL:MC:MD:CND=RM03:RP0456.011,CZRJBB.P11
 RUN-TIME:  22 14  .7 SECONDS
 RUN-TIME RATIO: 777/37=20.7
 CORE USED:  36K  (71 PAGES)
```